공학박사학위논문

# RDB to Semantic Data Transformation for Semantic Data Publication and Utilization

## RDB to 의미 데이터 변환기법에 기반한 의미 데이터 생성 및 활용 방법

**2017년 2월**

서울대학교 대학원

전기·컴퓨터공학부

전 희 국

# RDB to Semantic Data Transformation for Semantic Data Publication and Utilization

## RDB to RDF 변환기법에 기반한 의미 데이터 생성 및 활용 방법

지도 교수 김 형 주

이 논문을 공학박사 학위논문으로 제출함

2016년 10월

서울대학교 대학원
전기·컴퓨터공학부
전 희 국

전희국의 공학박사 학위논문을 인준함

2016년 12월

위 원 장 이 상 구 (인)

부위원장 김 형 주 (인)

위 원 김 홍 기 (인)

위 원 심 규 석 (인)

위 원 임 동 혁 (인)

# Abstract

RDB to RDF transformation is a semantic information extraction method that supports the Semantic Web. The direct mapping, one of the RDB to RDF transformation methods, is a representative mapping method recommended by the W3C. The direct mapping processes an automatic mapping from relational data to RDF data. Semantics preservation is an important property of the direct mapping to transform relational data to semantic data without information loss. However, existing direct mapping methods have problems that violate semantics preservation in specific cases. To comply with the semantics preservation, a hierarchical direct mapping method is provided. Rules of the hierarchical direct mapping are defined based on lemmas that represent features of semantic data transformation. A hierarchical semantic vocabulary is also defined to generate sound and precise semantic data. Next, this thesis also focused on developing an effective direct mapping to generate lightweight and intuitive semantic output data. Thus, the optimized hierarchical direct mapping is provided based on a relational meta-schema vocabulary. Rules of multi-column keys are defined to reduce repetitive constraint data generation problems. Rules for multiple keys are also defined because relational tables may contain multiple foreign keys or unique constraints that affect the output data size. The relational meta-schema vocabulary describes concepts of relational data and relationships among the concepts. The optimized hierarchical mapping method uses initially defined relational concepts from the

vocabulary, and generates compact and intuitive semantic output data. Finally, a semantic metadata based information retrieval method is provided as semantic data utilization. Existing ranking methods do not have direct methods of evaluating the meaning of links. In this thesis, a semantic metadata based ranking approach is proposed to directly analyze the meaning of links by using a semantic Web data structure. The semantic Web data structure is built upon semantic metadata extracted from the Web data by using the RDB to RDF transformation method described above. The provided method evaluates the weight of the links for stratifying rank values based on their importance in the semantic Web data structure. The experimental results showed that the proposed mapping method performs semantics preserving RDB to RDF transformation and outputs smaller size semantic data with better quality, and the weighted semantic metadata based ranking approach outperforms existing methods.

**Keywords:** Semantic Web, Relational Database, RDB2RDF, RDF, RDFS, OWL, Semantic Information Retrieval

**Student Number:** 2011-30253

# Contents

**Contents**

# List of Figures

## List of Figures

## List of Figures

# List of Tables

# Chapter 1
# Introduction

## 1.1 Research Motivation

As the World Wide Web produces a greater amount of information over time, such information needs to be processed more effectively and efficiently to provide more accurate information. In 1997, Resource Description Framework (RDF) [1] has been provided by the World Wide Web Consortium to generate semantic data. RDF has become a W3C recommendation in 1999 and has been studied to improve the semantic Web. In addition, other semantic data models such as RDFS (RDF Schema) [2], OWL (Web Ontology Language) [3] are developed to generate more detailed semantic metadata.

Semantic data can be published by modeling real world objects using semantic languages. However, as the majority of data on the Web is published

from relational databases [4], transforming knowledge from relational data to semantic data has become one of the major issues in the field of semantic data publication [5, 6].

Relational databases to RDF (RDB2RDF) is one of the information extraction methods that support the Semantic Web. In 1998, Tim Berners-Lee proposed the concept of mapping relational databases to the Semantic Web [7]. Afterwards, many approaches have been studied to improve mapping relational databases to RDF. Additionally, the W3C has organized a working group to standardize technologies related to RDB2RDF.

The direct mapping [8] is a representative mapping method recommended by the W3C to support automatic mapping of relational data to RDF data. Figure 1.1 illustrates an example of direct mapping, which defines mapping rules to transform both relational schema and instance data to RDF data. In the field of direct mapping, researchers have studied effective automatic processes focused on semantics preservation. Semantics preservation of the direct mapping is the reflection of relational integrity constraints within the mapping result [9, 10]. As integrity constraints define the semantics of the database, mapping with integrity constraints generates more semantically accurate results. However, although transforming relational data using integrity constraints has been studied, existing methods still lack support for the transformation of all integrity constraints. Moreover, we observed that incorrect semantic data generation problems can occur in specific cases. Thus, an improved direct mapping method is proposed in this

thesis. The method supports all integrity constraints to provide semantic preserving RDB to RDF transformation.

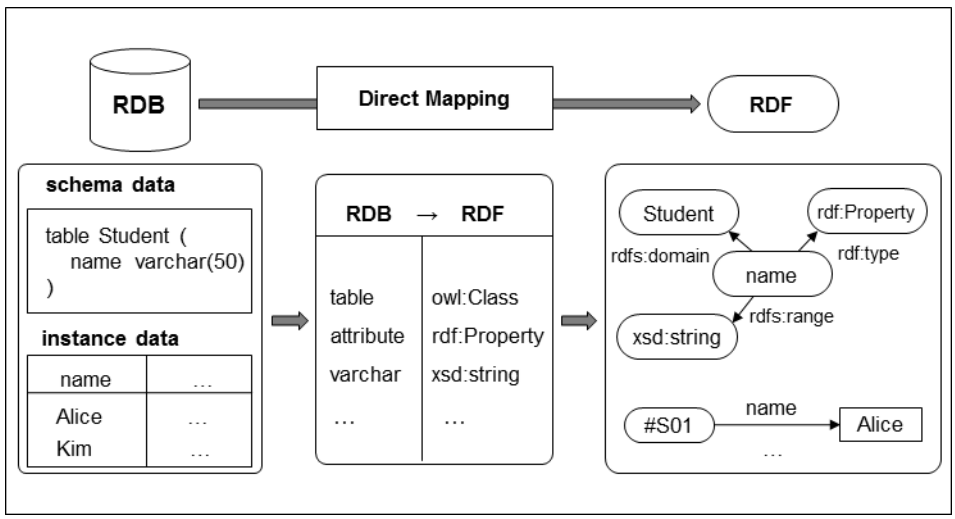



**Figure 1.1.** A simplified example of direct mapping.

In this doctoral dissertation, we propose semantics preserving relational databases to RDF transformation to achieve semantic data publication (Figure 1.2(a)). The proposed transformation method provides semantics preserving mapping rule without information loss or incorrect semantic data generation. We adapt a direct mapping to transform relational data into semantic data automatically and improve the mapping method using hierarchical semantic vocabulary. In addition, we propose a semantic data utilization method based on the generated data using the proposed transformation method (Figure 1.2(b)). We also developed a link-based information retrieval method using semantic data. The information

retrieval method evaluates Web data based on the importance of the semantic resources.

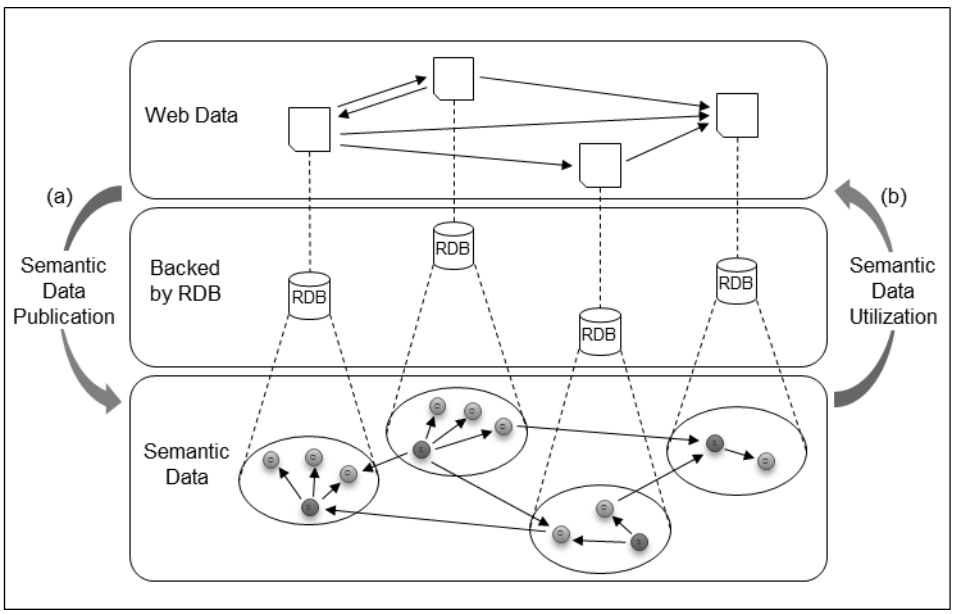



**Figure 1.2.** The purpose of the dissertation.

# 1.2 Research Contributions

In this section, the research contributions of this thesis for RDB to RDF transformation based semantic data publication and utilization are presented. The detailed contributions are discussed in research topics: semantics preserving RDB

to RDF transformation, optimization of processing RDB to RDF transformation, and information retrieval using generated semantic metadata.

### 1.2.1 Semantic Preserving Hierarchical Direct Mapping for RDB to RDF transformation

The problem in the existing direct mapping methods, which are automatic RDB to RDF transformation methods, is that it does not fully support mapping integrity constraints. Semantics preservation of direct mapping depends on the quality of transforming integrity constraints of relational input data. This thesis provides observations of specific cases in which semantic information loss or incorrect semantic data generation problems occur. The definition of semantics preservation is improved to evaluate the accuracy of the RDB to RDF mapping methods. A hierarchical structured semantic vocabulary is also defined to be utilized in direct mapping rules. The mapping rules comprise general mapping rules and constraint mapping rules. The general mapping rules are for mapping relations, attributes, and other general relational objects. The general mapping rules are defined to avoid semantic information loss during transformation of general relational objects. The constraint mapping rules are for mapping integrity constraints and are defined to reduce incorrect semantic data generation. Finally, the semantics preserving direct mapping method is implemented, and a comparative experimental study is done with both synthetic and real datasets. The experiments show that the proposed mapping method performs semantics preserving RDB to RDF transformation and

generates semantically accurate results.

The contributions are summarized as follows:

- Mapping rules to solve the incorrect semantic data generation problems are designed. The rules are defined to ensure semantics preservation, a fundamental property of direct mapping.
- A hierarchical semantic vocabulary to represent relational schemas is defined; mapping processes can be simplified by using the vocabulary.
- The scope of the semantics preservation is extended; in particular, the inverse transformation of output semantic data should identical to the original input data.
- A direct mapping system using the mapping rules and the semantic vocabulary is implemented. The system uses the MapReduce framework to manage large scale relational data on the Web.

### 1.2.2 Optimized Hierarchical Direct Mapping using Relational Meta-schema Vocabulary

Relational attributes can be assigned with one or more integrity constraints. As a result, transformed data by direct mapping methods contains repetitive identical constraint sub-graphs. Output semantic data size is generally greater than input relational data size due to data structure of semantic data, which is an RDF graph data structure. Thus, this thesis provides an optimized direct mapping method to

reduce output data size by eliminating repetitive integrity constraint data generation processes. First, multi-column keys, which comprise two or more attributes, are integrity constraints that contain repetitive constraint assertions. The provided method generates a single constraint semantic sub-graph and transforms each attribute to be inherited by the generated sub-graph. Second, two or more foreign keys or unique constraints can be defined in one table. As multiple foreign keys or multiple unique constraints are the most repetitive constraint data generation factor, this thesis provides a more optimized method to reduce data size during mapping multiple key constraints. Finally, a relational meta-schema vocabulary is designed to initially define relational concepts and relationships among relational concepts. The mapping rules that use the relational meta-schema vocabulary can generate compact and intuitive semantic relational output data. The thesis also provides optimized mapping rules that adapt the relational meta-schema vocabulary. The experimental results show that the proposed mapping method generates fewer semantic data than the previous approaches, and outputs RDF data that preserves the semantics of original input data.

The contributions can be summarized as follows:

- Rules for mapping multi-column primary key, foreign key, and unique constraints are provided. The proposed method based on the rules transforms relational data into semantic data with lessor data size.

- Rules for mapping multiple foreign keys and multiple unique constraints are provided to reduce output semantic data without semantic information loss.
- A semantic relational meta-schema vocabulary is defined. The vocabulary defines relational tables, attributes, and integrity constraints in semantic notations. The vocabulary also defines relational resources with a more hierarchical structure to encapsulate complicate structures of constraint information and to provide a lightweight and intuitive expression for mapping relational data.
- Optimized mapping rules that adapt the relational meta-schema vocabulary are defined. The mapping rules are optimized to reduce repetitive constraint data and generate compact output data with fewer resources.

### 1.2.3 Semantic Data Utilization to improve Web Information Retrieval

To improve the performance of Web information retrieval [11-13], this thesis utilizes semantic data. PageRank is a representative link-based information retrieval method [14, 15]. In a Web structure, hyperlinks among Web documents do not contain explicit linking information. Thus, ranking methods assume that Web pages referenced with many in-links are important pages. This thesis provides a method to evaluate the importance of Web pages by semantics of the pages. Every semantic resources in Web pages are evaluated and assigned importance values by

the provided method. Therefore, Web pages can be assessed by using importance values of semantic resources that are contained in Web pages.

The contribution of this paper can be summarized into threefold. First, a ranking method based on semantic information to generate more accurate ranking results is proposed. The method considers the meaning of pages and links by evaluating their semantic information in a semantic-link-based data structure, rather than using the number of links among the pages. Second, an algorithm to reduce the probability of giving high rank values to unimportant pages is designed. Using semantic information instead of hyper-links, the method is able to calculate rank values based on the semantics of the pages. Thus, the method guarantees that highly ranked pages contain valuable information. Finally, a framework that transforms a hyperlink-based Web structure into a semantic Web data structure is implemented, and the page importance values are evaluated based on the structure.

## 1.3 Outline

The remainder of the thesis is organized as follows. Chapter 2 introduces preliminaries and related work to describe researches of relational data, semantic data, semantic information extraction, RDB to RDB transformation, and information retrieval.

In Chapter 3, a semantics preserving hierarchical direct mapping for RDB to RDF transformation is described. The hierarchical direct mapping method is developed to ensure the semantics preservation of the mapping. Mapping rules are defined based on lemmas that represent features of semantic data transformation. A hierarchical semantic vocabulary is also defined to generate sound and precise semantic data.

In Chapter 4, an optimized hierarchical direct mapping method is presented. The mapping method is designed to reduce repetitive data during mapping integrity constraints. As repetitive semantic data is majorly generated by mapping multi-column keys and multiple keys, optimized mapping rules for mapping multi-column keys and multiple keys are developed. A relational meta-schema vocabulary, which is used by the optimized mapping rules, is also defined to reduce repetitive data and to output compact and intuitive semantic data.

In Chapter 5, a framework for semantic metadata based Web information retrieval method is described. The method evaluates semantic resources that are contained in Web pages and calculates importance value of each semantic resource. As a result, pages are evaluated by the importance values of semantic resources that each page contains.

Finally, the conclusions of the thesis and future research are presented in Chapter 6.

# Chapter 2
# Preliminaries

## 2.1 RDF

Semantic Web enables people to generate semantic data, vocabularies, or rules. Semantic Web is a vision of the linked data on the Web. The goal of Web of data is to enable computers to process effective and interoperable work on the Web [16]. The Resource Description Framework (RDF) is a representative data model to represent metadata of Web resources [17]. The RDF was published in 1997 and recommended in 1999 by W3C. The RDF data is modeled based on a graph structure, in which a node is a semantic resource and an edge is a relationship between two resources. The RDF graph structure comprises a set of triples (Figure 2.1). Each triple consists of a subject, property, and object. In an RDF graph, nodes are subjects or objects, and edges are properties. URI (Uniform Resource Identifier)

[18] is used to uniquely identify semantic resources: subjects, properties, and objects. URI is a series of characters that becomes a thing's unique identifier. In addition, objects can be literals, which are leaf nodes that cannot be subjects. Objects can also be blank nodes, which is an RDF node without URIs or literal values.

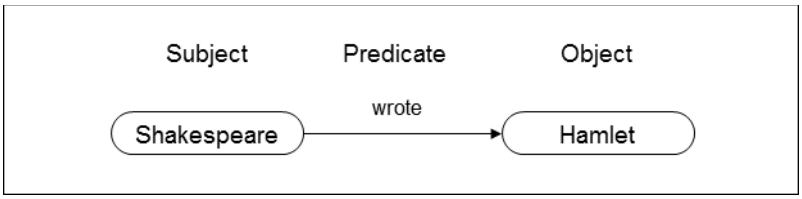



**Figure 2.1.** An example of an RDF triple.

The following definition is a formalization of RDF graphs [19, 20]:

**RDF triple:** Suppose $U$ is an infinite set of RDF URI references, B is an infinite set of blank nodes, and L is an infinite set of RDF literals, then a triple ($s$, $p$, $o$) $\in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple. In such a triple, $s$ is called the subject, $p$ the predicate, and $o$ the object.

Figure 2.2 shows different notations for the same RDF graph. RDF/XML [21] is the first normative syntax of RDF data provided in 1999. In 2008, Notation3 (N3) is developed by Tim Berners-Lee [22]. N3 is a non-XML serialization of RDF models, and more compact and readable than the XML/RDF syntax. In 2011, Terse RDF Triple Language (Turtle) is developed by Dave Beckett, which is a subset of

N3 [23]. Turtle does not rely on XML. Therefore, Turtle is generally recognized as a readable syntax, and is easier to edit manually than the RDF/XML syntax. Turtle is used for the syntax of SPARQL [24] to express query patterns. R2RML [25] also adopts Turtle syntax to transform relational databases into RDF graphs.

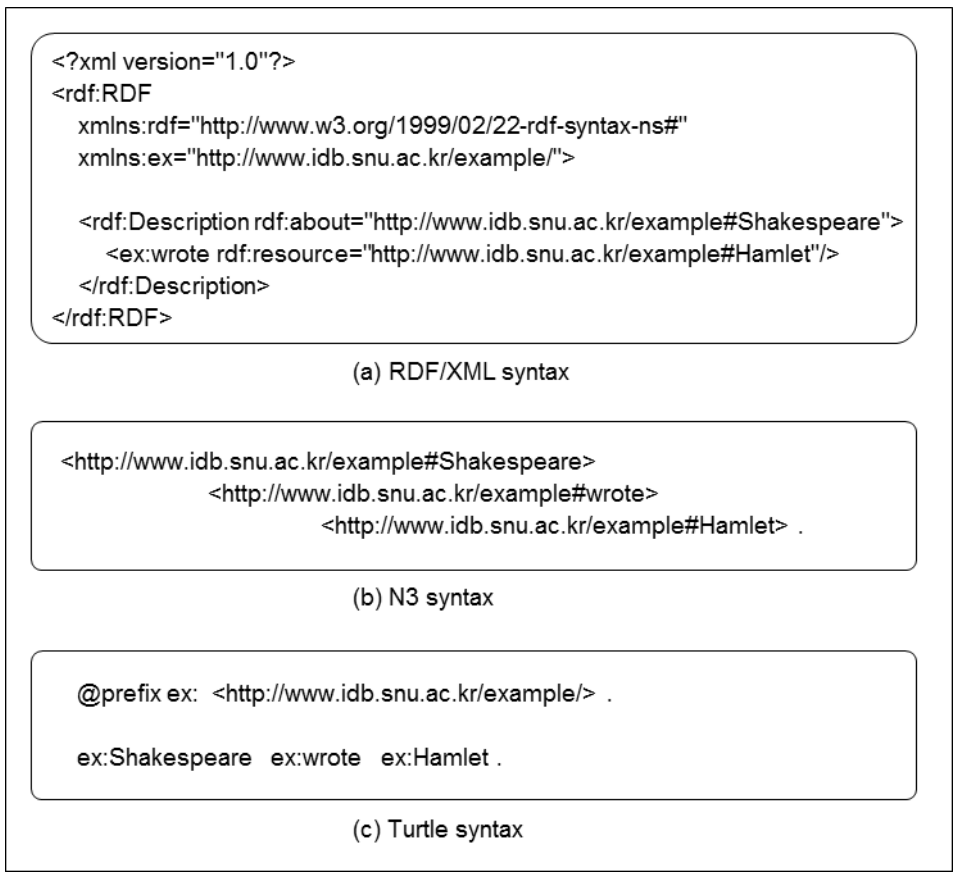



**Figure 2.2.** An example of RDF notations.

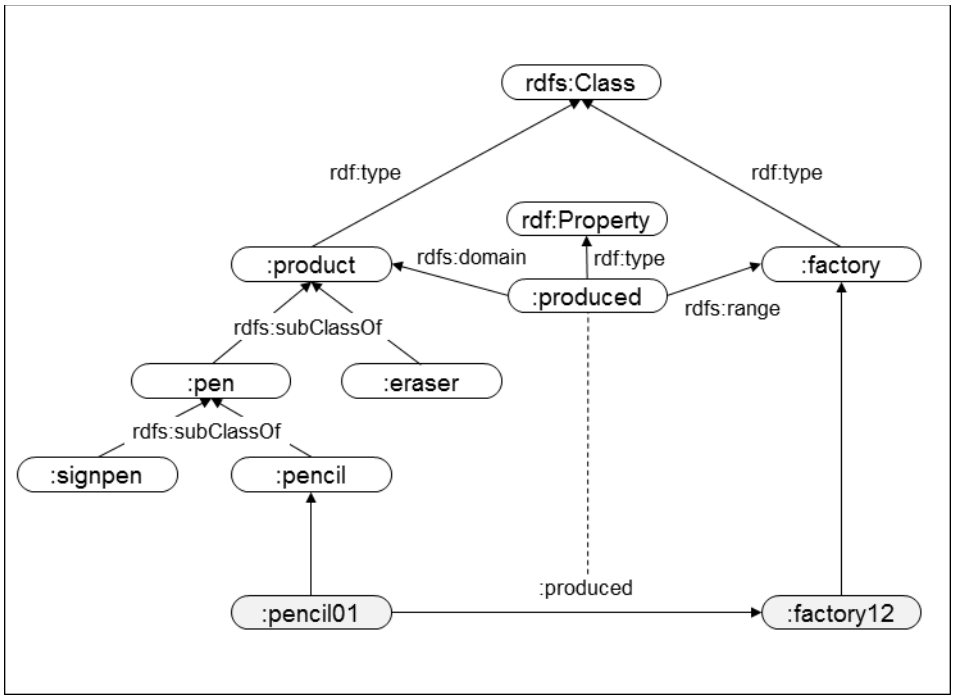



**Figure 2.3.** An example description of RDFS.

# 2.2 RDFS

RDFS (Resource Description Framework Schema) is an extension of RDF vocabulary to describe classes, properties, and utility properties [2]. The RDFS was published in 1998 and recommended in 2004 by W3C. Vocabularies of Classes are Resource, Class, Literal, Datatype, XMLLiteral, and Property. Vocabularies of Properties are domain, range, type, subClassOf, subPropertyOf, label, and

comment. Furthermore, instances of RDF property seeAlso and isDefinedBy are for the utility properties. Figure 2.3 is an example of data described by RDFS.

## 2.3 RDFa

A semantic markup language is used to embed metadata in Web documents. RDFa [26], Microformats [27], and Microdata [28] are representative semantic markup languages. RDFa is derived from the RDF data model and recommended by W3C in 2015. RDFa provides high applicability in the Web environment and enables Web documents to contain semantic metadata in RDF data. Figure 2.4 shows an example of using RDFa to annotate RDF metadata in XHTML code. Web documents written in XHTML code with RDFa are viewed as Web pages in Web browsers; moreover, the documents can be used as semantic metadata by adopting RDFa parsers.

XHTML with RDFa

```
<div about="http://.../GreatAdventure">
    <h3 property="dc:creator">John</h3>
</div>
```

**Figure 2.4.** An example of RDFa annotation.

Semantic metadata management has been developed in various fields. Google, Microsoft, and Yahoo! established schema.org in 2011 for precise search results using RDFa, Microformats, and Microdata. Google's Rich Snippet and Yahoo!'s BOSS (Build your Own Search Service) are technology that use semantic metadata in search results [29]. Facebook's Open Graph protocol is for semantic metadata in a social network [30]. Drupal and Wordpress, major content management systems, provide an automatic semantic tagging module [31]. Various RDFa-related methods as well as RDFa annotation systems [32-35] have also been developed. W3C has provided a distiller and a parser for RDFa. RDFauthor [36] is an integrative RDFa management framework.

## 2.4 OWL

The Web Ontology Language (OWL) [3] is based on description logic and a syntactic extension of RDF and RDFS. The OWL was published in 2000 and recommended in 2004 by W3C. In the initial OWL specification, the W3C identified three particular variants of OWL: OWL-Lite, OWL-DL, and OWL-Full. OWL Full is an unrestricted dialect of OWL with all constructs used in any combination. OWL-DL is a dialect of OWL restricted to ensure decidability, in which all constructs are allowed but certain restrictions exist. OWL-Lite is a subset of OWL-DL designed to encourage early adoption and is easier to implement. For instance, Table 2.1 describes a list of semantic triple data using OWL notation.

**Table 2.1.** Examples of semantic triple data using OWL notation.

| ( Subject , Property , Object ) | Types of property | Semantics |
|---|---|---|
| (Professor , teaches , Student) | Object property | Relation between resources |
| (Person , age , "23" ) | Data type property | Subject has a data type value |
| (Product , produced , Factory) | Functional property (FP) | N:1 relationship |
| (Shakespeare , writes , Book ) | Inverse functional property (IFP) | 1:N relationship |
| (Husband , , marriage , Wife ) | FP and IFP | 1:1 relationship |

In 2009, the OWL 2 Web Ontology Language (OWL2) is recommended by W3C to be used for the formally defined meaning [37]. Figure 2.5 shows relationships among OWL 1, OWL2, and RDFS. OWL 2 specifies three profiles (often called as fragments in logic): OWL 2 EL, OWL 2 QL, and OWL 2 RL. OWL 2 EL is for applications of ontologies, which contain large numbers of class and properties. OWL 2 QL is for sound and complete query answering. OWL 2 RL is for applications, which require scalable reasoning with less expressive power.

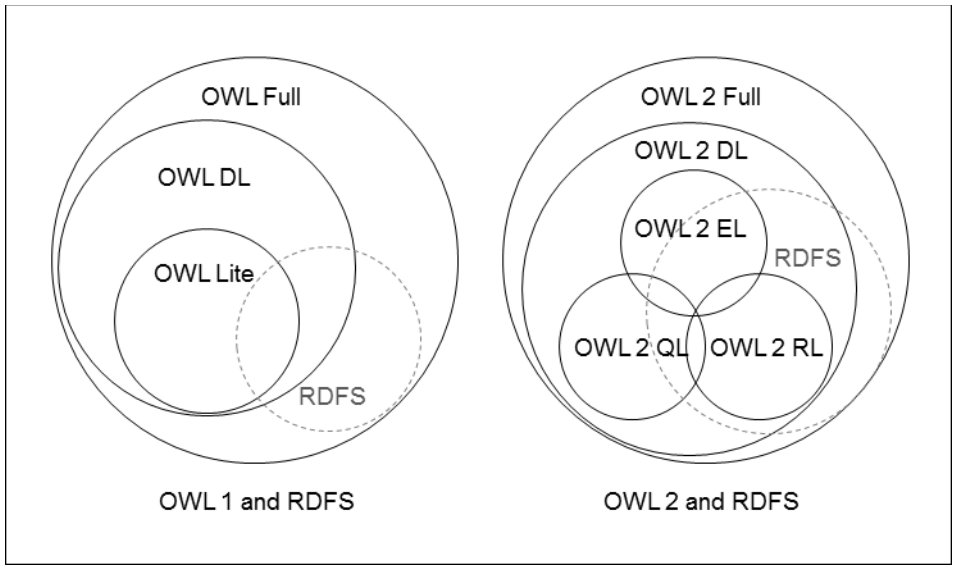



**Figure 2.5.** OWL syntactic subsets.

# 2.5 RDB to RDF Transformation

RDB2RDF is a mapping method that transforms relational data to semantic data represented by Resource Description Framework (RDF), which is a machine readable formant. The RDF data model [38] is a language that describes semantic information on the Semantic Web. The basic unit of RDF data is based on a graph structure called (subject, property, object) triple [39-41]. Thus, RDF is more flexible and interoperable to publish information on the Web relative to the relational data model. However, as nearly 70 percent of web sites are backed up by relational databases [4], it is required to utilize existing relational data with the

RDB2RDF methodology for the improvement of the Semantic Web.

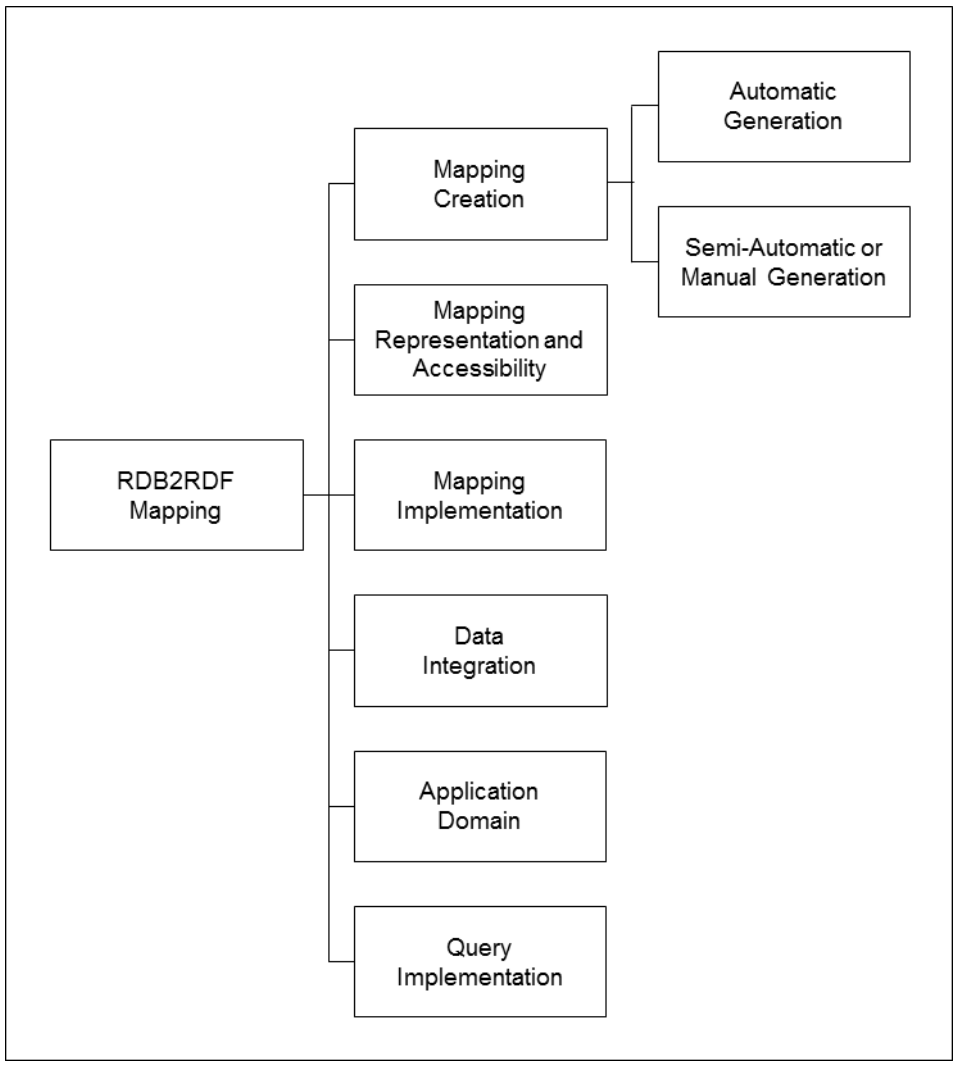



**Figure 2.6** The RDB2RDF mapping approaches.

The RDB2RDF mapping approaches are classified into six categories [42]: mapping creation, mapping representation and accessibility, mapping implementation, query implementation, application domain, and data integration. Mapping creation is a generation method of mapping between relational databases and RDF. Mapping representation and accessibility is a method to represent and access mapping between relational databases and RDF on the Web. Mapping implementation can be performed by either an Extract Transform Load (ETL) system or a query-driven dynamic implementation. The ETL system transforms relational data into semantic data and stores the semantic data using RDF repository. On the other hand, a query-driven dynamic implementation transforms relational data into semantic data during a query execution process. Query implementation is a method to process input queries of a RDB to RDF system by using SPARQL, SQL, or both SPARQL and SQL. Application domain is a category that comprises various applications using RDB2RDF approaches or used for RDB2RDF approaches. Data integration is a method to integrate generated heterogeneous semantic output datasets using URI defined semantic resources.

Mapping creation, which is one of the RDB2RDF mapping approaches, is studied to improve the generation of the mappings between RDB and RDF. Mapping creation can be performed either automatically, semi-automatically, or manually [43]. Domain semantics-driven mapping is the manual (or semi-automatic) mapping method [44, 45]. The W3C RDB2RDF working group recommended the R2RML mapping language [46, 47] for users to customize mappings. Mapping tools, such as D2RQ [48, 49], Virtuoso [50, 51], Ultrawrap [52,

53] and others, have also been provided to support manual mapping. Manual mapping can be used when RDB data is comparatively small, or domains of RDB require specific background knowledge to be transformed. On the other hand, direct mapping is an automatic mapping method, published by the W3C RDB2RDF working group in 2012 [54]. The direct mapping utilizes relational database instances and schemas as input and generates RDF semantic data automatically. The direct mapping can be used when RDB data is large scale or contains complex structure for human to intuitively understand the data.

This thesis is focused on mapping creation to build efficient and effective RDB to RDF transformation without information loss. For this purpose, related works on mapping creation are surveyed and provided in the next sections. A representative manual mapping language, R2RML, is described in section 2.5.1. Direct mapping is described in section 2.5.2 and fundamental properties of direct mapping are provided in section 2.5.3. Finally, a discussion on the semantics preserving direct mapping is provided in section 2.5.4.

### 2.5.1 R2RML

R2RML (RDB to RDF Mapping Language) [46] is a mapping language to express customized mappings that transform relational data into RDF data. R2RML was developed in 2010 and recommended in 2012 by W3C RDB2RDF working group. Domain semantics-driven mapping, also known as manual mapping, can be processed by using R2RML that specifies relationships between relational data and

RDF data.

In R2RML, Turtle syntax is used to represent RDF graphs, and Turtle syntax is also used to define a mapping description of a relational data to be transformed into semantic data. In a RDB to RDF mapping process using R2RML, a relational table is set as logical table, a Subject Map specifies a value of relational instance to be used an RDF subject resource, an attribute is mapped by a Predicate Map, and an attribute value is mapped by an Object Map (Figure 2.7).

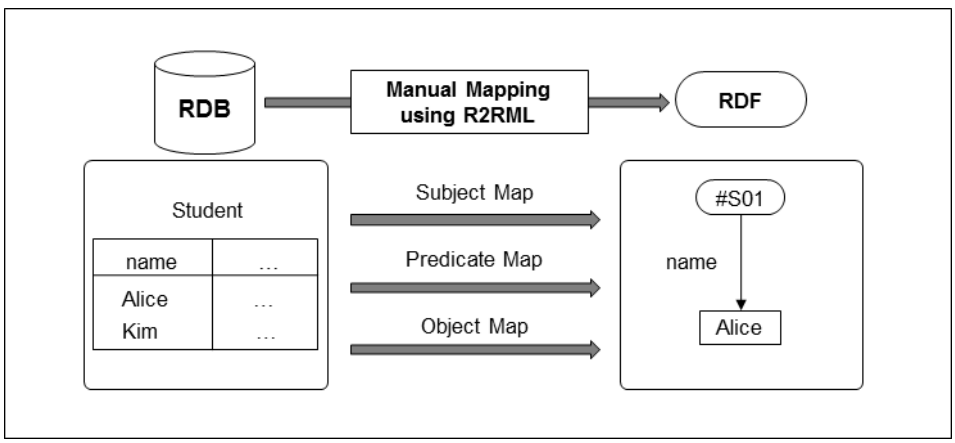



**Figure 2.7.** Overview of RDB2RDF transformation using R2RML.

Figure 2.8 and Figure 2.9 show the R2RML data model [55]. The model provides resources named rr:TriplesMap, rr:LogicalTable, rr:SubjectMap, rr:TermMap, rr:PredicateObjectMap, and rr:RefObjectMap. The model also provides predicates named, rr:logicalTable, rr:subjectMap, rr:class, rr:predicateObjectMap, rr:predicateMap, and rr:objectMap. The term "rr:" is a

Turtle prefix referencing the URI of R2RML <http://www.w3.org/ns/r2rml#>. The term "rr:" has been omitted hereafter for readability.

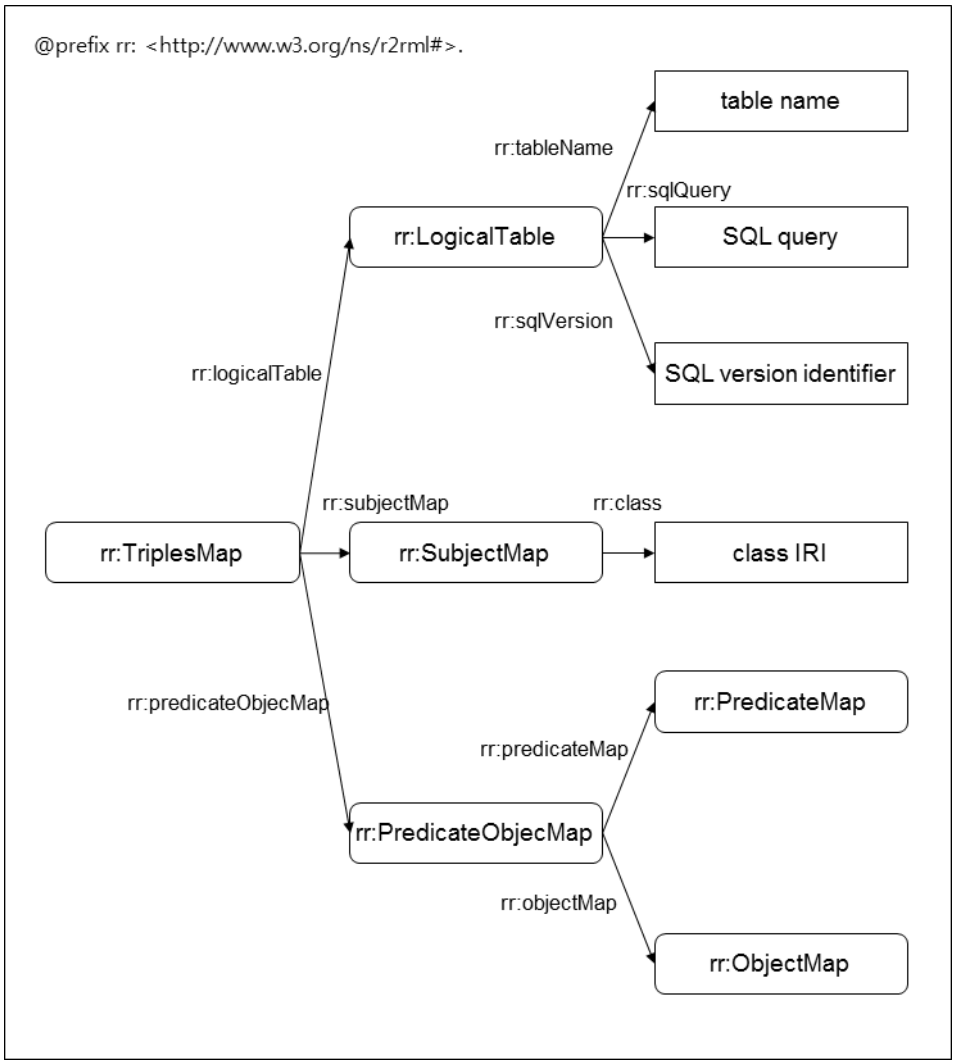



**Figure 2.8.** R2RML data model.

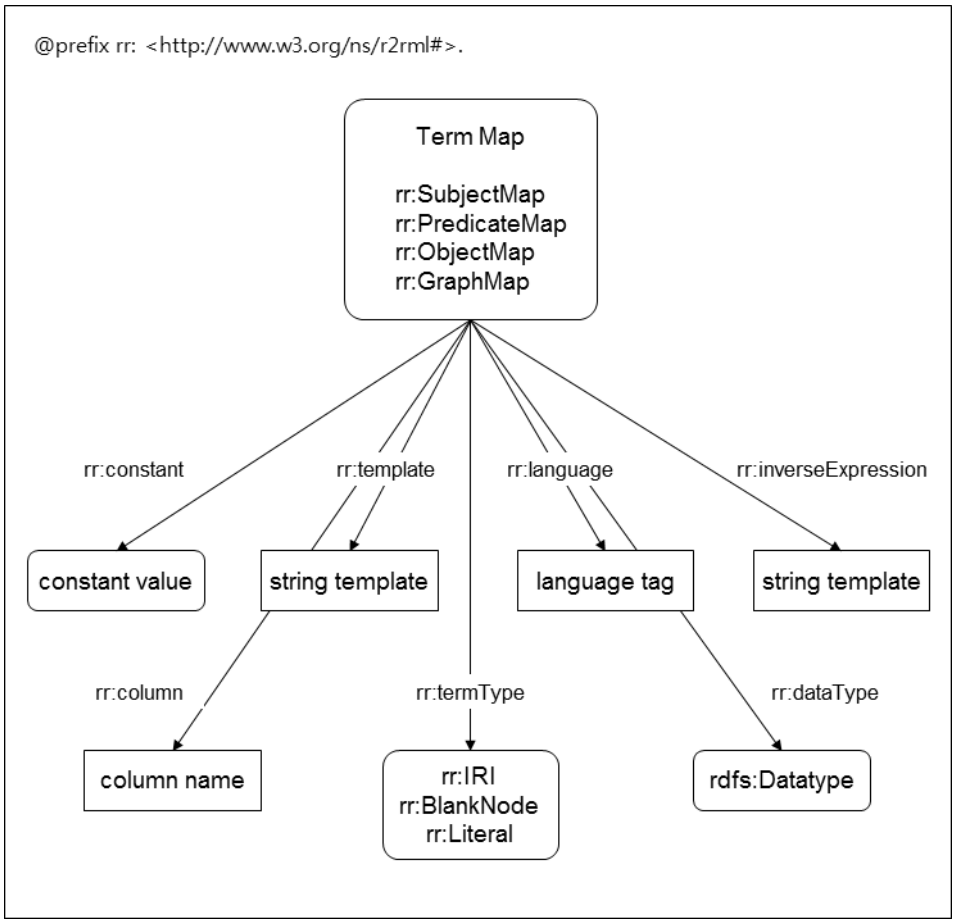



**Figure 2.9.** R2RML data model: Term Map.

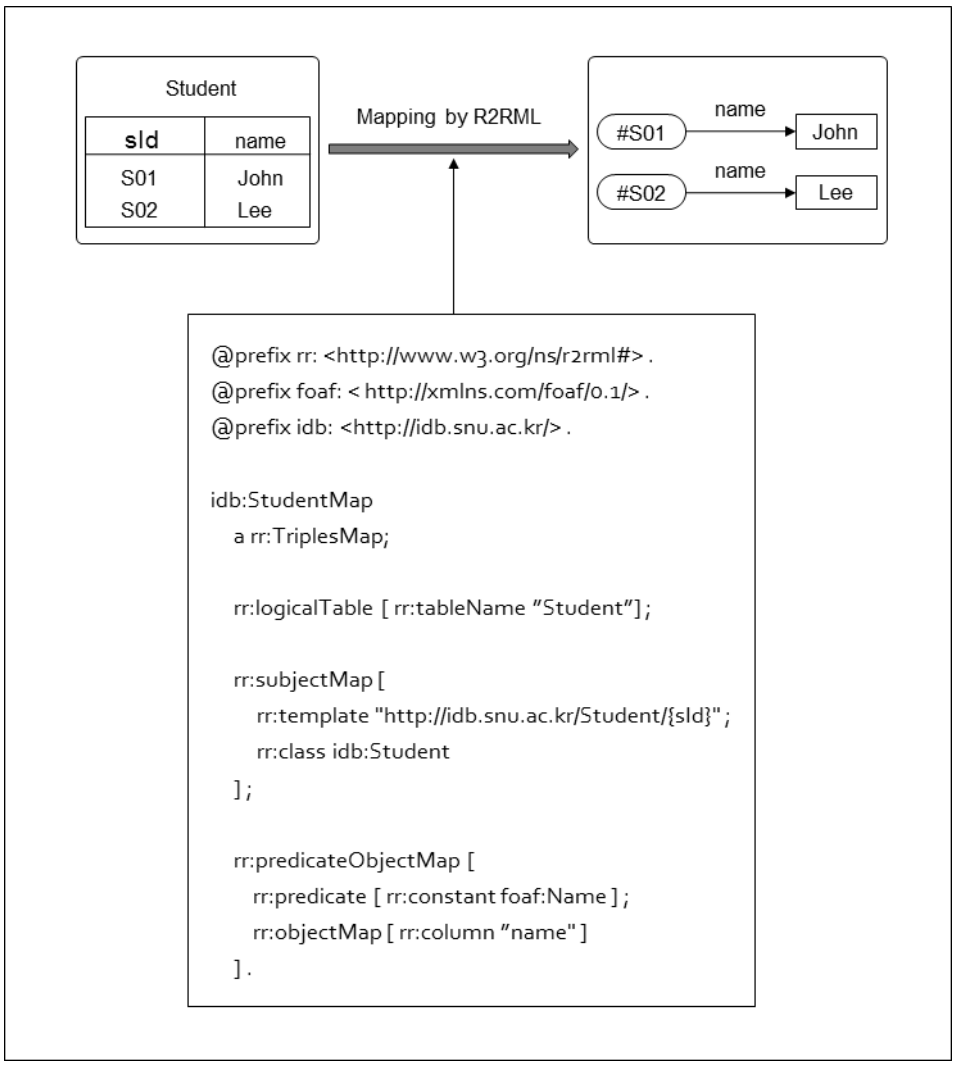



**Figure 2.10.** An example of mapping by using R2RML.

Figure 2.10 shows an example of mapping by using R2RML. An R2RML mapping is able to contain a set of TriplesMaps. A TripleMap specifies mapping information about a LogicalTable, which denotes a relational table, view, or a result of a SQL query that is used to retrieve a specific set of relational data. Each TripleMap contains a single SubjectMap, which denotes subject terms of a given LogicalTable. To define values of subjects, SubjectMap as a template-valued TermMap is used to create URIs of subjects based on a template form ("http://idb.snu.ac.kr/Student/{sId}" in Figure 2.10 as an example). Each TripleMap also contains multiple PredicateObjectMaps, which denote predicates and objects. PredicateMap is used in a PredicateObjectMap to describe predicates. PredicateMap as a constant-valued TermMap is used to generate all the same specified predicate values for the PredicateMap. ObjectMap is another used resource in a PredicateObjectMap to describe objects. ObjectMap as a column-valued TermMap is used to generate values in the specified attributes in the LogicalTable.

R2RML is the most expressive mapping language to process a RDB2RDF transformation and widely used in RDB2RDF applications [56]. On the other hand, as R2RML is a language used for RDB2RDF transformation manually, a mapping for large-scale relational databases relatively expensive process than an automatic mapping methods.

## 2.5.2 Direct Mapping

Direct mapping [54] is an automatic mapping creation method, which transforms relational input data including schema data into RDF graph data called the direct graph. Direct mapping was developed in 2010 and recommended in 2012 by W3C RDB2RDF working group.

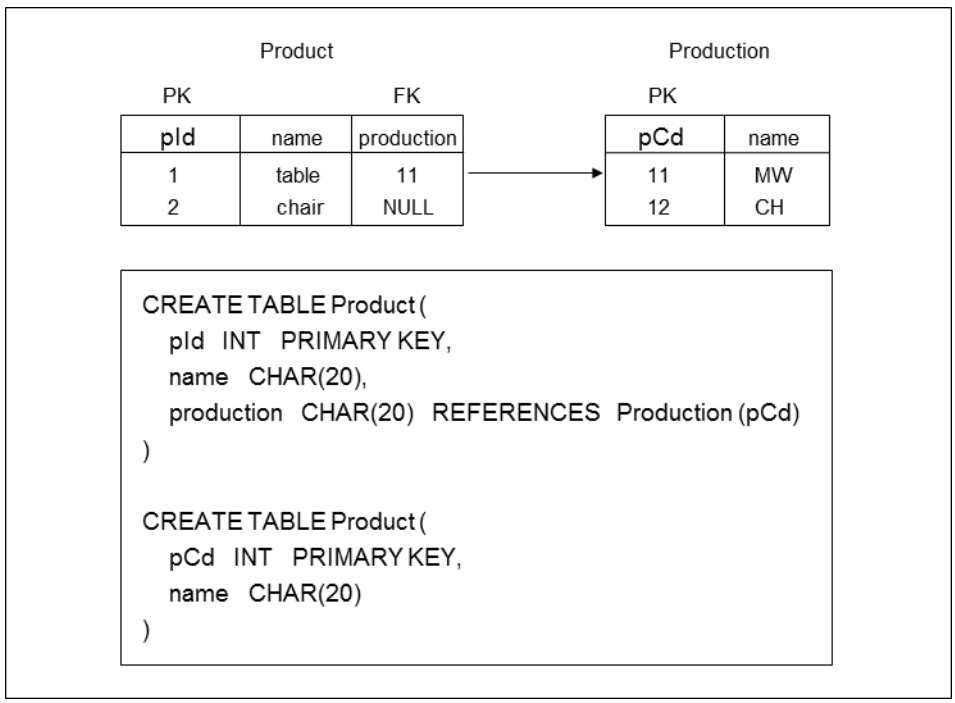



**Figure 2.11.** An Example input relational data of the direct mapping.

Direct mapping can be viewed as a function from relational data with integrity constraints to semantic data. Figure 2.11 describes an example relational input data of the direct mapping. A table named *Product* contains an attribute *sId* as a primary

key, an attribute *name*, and an attribute *production* as a foreign key referencing table *Production*. The other table *Production* contains an attribute *pCd* as a primary key and an attribute *name*. Figure 2.12 describes the result of the direct mapping from the input data in Figure 2.11. The output graph comprises a set of RDF triples. Suppose that the base IRI of output data is <http://idb.snu.ac.kr/example/>. Primary key attributes with base IRI are used to generate subject resource. Two resources typed *Product* and two resources typed *Production* are generated. Predicates are generated from attribute names of relational tables, and objects are generated from attribute values.

In the direct mapping, null values are not involved in the RDB2RDF transformation process. Note that product with *pId* value 2 has null value for attribute *production* in Figure 2.11. As a result, the output of the product does not have triples of foreign key reference information because the direct mapping does not generate triples for null values (Figure 2.12). However, if every instances of *Product* have null values for attribute *production*, then the output direct graph cannot express that attribute *production* is a column of the table *Product*. Authors of [57] provide the use of blank node as existential variables for null values in database. On the other hand, [58] surveyed with 10 participants, who have contributions to a current LOD (Linked Open Data) dataset. The result of the survey shows that a triple (subject, property, blank node as an object) is mostly understood as "the subject has a value but its value is hidden." or "I would not publish such a triple." Therefore, transforming schemas of relational data is a considerable issue in the direct mapping to prevent information loss.

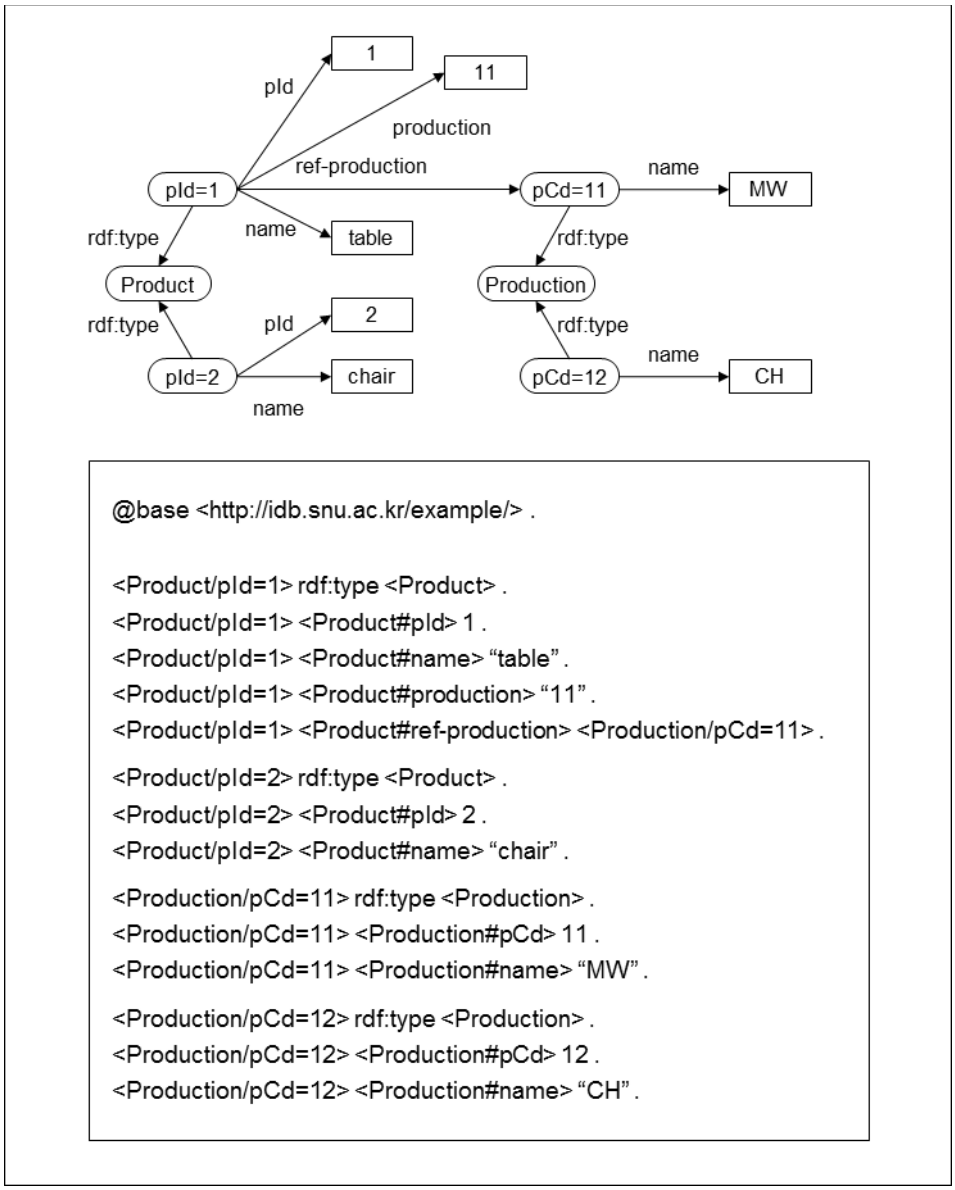



**Figure 2.12.** An example result of the direct mapping.

To transform a relational schema into semantic data, this thesis referenced previous works [59-61] for the following assumptions. First, the relational schema is available in SQL DDL, which is the most accurate form for the relational database structure. Second, the relational schema is at least up to third normal form. If a schema is unnormalized, then the schema can be algorithmically transformed to third normal form.

### 2.5.3 Properties of Direct Mapping

Information preservation, query preservation are fundamental properties of the direct mapping. Monotonicity and semantics preservation are desirable properties of the direct mapping [10]. If all the relational instance data can be transformed into semantic data by a direct mapping method, then the direct mapping method is information preserving. Showing that the direct mapping is information preserving is straightforward, we can recover the original input instance data from output semantic data by an inverse-mapping. If every SQL queries of a given relational database can be transformed into equivalent queries based on the output semantic data, then the direct mapping is query preserving. If newly inserted data of a relational database can be reflected in the output data, then the direct mapping is monotone. Finally, if key constraints of a relational database can be transformed into the output data, then the direct mapping is semantics preserving. However, monotonicity and semantics preservation are properties that conflict with each other. It is proved that no monotone direct mapping is semantics preserving if foreign keys are considered [9]. Thus, if a direct mapping method is semantics

preserving, then the method is a non-monotone direct mapping.

### 2.5.4 Semantics Preservation of Direct Mapping

Further research has been conducted to transform semantic data from relational data without information loss [59, 62-68]. RDFS (RDF Schema) and OWL (Web Ontology Language) are utilized for a more accurate mapping. The concept of RDF data can be modeled by RDFS or OWL in a manner similar to defining relational schema using SQL DDL (Data Definition Language). Moreover, as OWL contains more expressive semantic vocabularies, the mapping methods are able to express semantics of relational integrity constraints using OWL.

However, the computational cost of a mapping method using OWL can be more expensive than others without OWL. Semantic data modeling using OWL should consider integration of other semantic data already existing in a semantic Web structure, and would contain complex semantic data structure.

W3C provided RDFS 3.0 that is an RDFS and a subset of OWL [88]. OWL in RDFS 3.0 contains six vocabularies of original OWL vocabularies. This feature allows efficient and scalable implementation of semantic data. Authors of [89] defined RDFS Plus that is an extension of RDFS and also a subset of OWL. RDFS Plus contains 10 OWL vocabularies. The vocabularies are categorized by three classes: basic constructs, sameness, and other constructs. Both RDFS 3.0 and RDFS Plus provide RDFS and a subset of OWL to support more effective

generation of semantic data with less complexity of full OWL language (Figure 2.13).

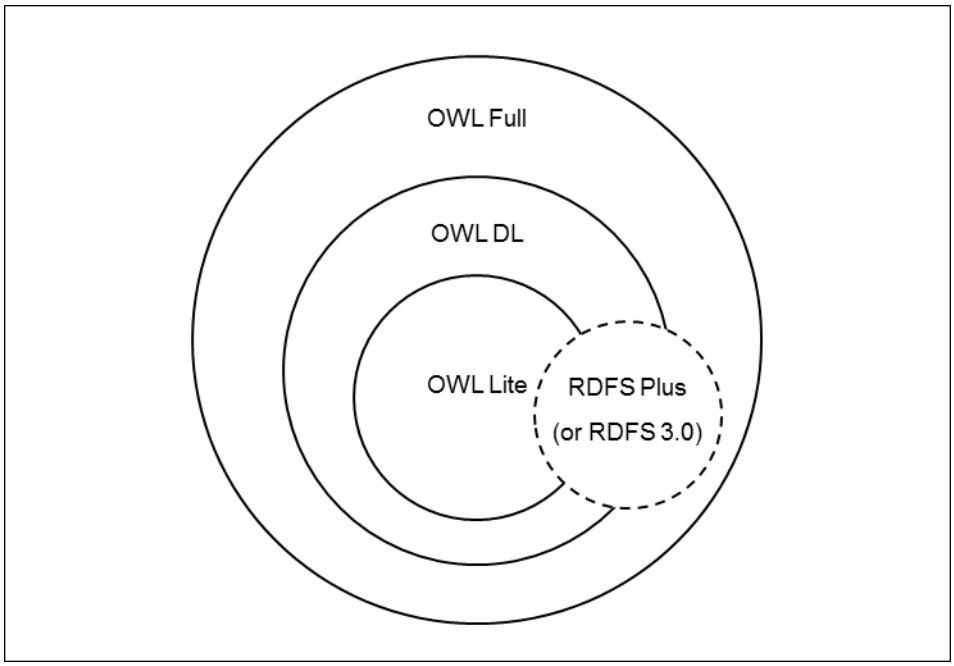



**Figure 2.13.** Layered structure of OWL and RDFS Plus (RDFS 3.0)

The authors of [9] proposed an augmented direct mapping that generates semantic data from integrity constraints of SQL DDL schema. As integrity constraints define the semantics of the relational databases, the quality of the augmented direct mapping depends on the transformation of integrity constraints of the relational databases. DB2OWL [65] and RDBToOnto [66] also provide augmented direct mapping tools, but these are restricted to support only referential integrity constraints. [67] utilizes the OWL language to process more rules, however, this

method still lacks support for the transformation of all integrity constraints by the SQL standard. Moreover, incorrect semantic data generation problems can occur in specific cases.

## 2.6 Terminologies

In this section, terminologies of the doctoral dissertation are described. RDB2RDF is an abbreviation of the relational databases to RDF transformation. Direct mapping is an automatic RDB to RDF transformation method. Semantics preserving direct mapping is a specific part of direct mapping methods that transforms integrity constraints into the mapping result. Monotone direct mapping [9] is also a specific part of direct mapping methods. To define a monotone direct mapping, assume that M is a monotone direct mapping, R is a relational schema, $\sum$ a primary and foreign key set, $I_1$ and $I_2$ are database instances of R, such that $I_1 \subseteq I_2$, then the monotone direct mapping is $M(R, \sum, I_1) \subseteq M(R, \sum, I_2)$. The equation means that a monotone direct mapping method should consider new data of a given database. Finally, semantic data is data defined by semantic languages such as RDF, RDFS, or OWL. Semantic metadata is semantic data annotated or embedded in Web documents by using RDFa.

# Chapter 3
# Semantics Preserving RDB to RDF Transformation

The direct mapping method of RDB to RDF transformation automatically generates semantic data from relational data for the pervasion of semantic Web data. However, existing direct mapping methods have problems that violate semantics preservation during mapping processes. In this chapter, a hierarchical direct mapping method is proposed that ensures the semantics preservation of the mapping. Mapping rules are defined based on lemmas that represent features of semantic data transformation. A hierarchical semantic vocabulary is also defined to generate sound and precise semantic data. The experiments show that the proposed method performs semantics preserving RDB to RDF transformation and generates semantically accurate results.

# 3.1 Motivation

Although existing RDB to RDF transformation methods using integrity constraints has been studied [59, 62-67], problems during direct mapping processes of previous approaches are still observed under specific conditions (Figure 3.1 as an example). Suppose there is an attribute with two or more integrity constraints (*not null* and *unique* in Figure 3.1(a)). In this case, the mapping result outputs a single RDF graph structure that combines all of the integrity constraints (a sub-graph rooted by *name* in Figure 3.1(b)). Therefore, a new sub-graph can be extracted from the merged RDF graph, which can misinterpret and generate a new constraint not in the original data (*primary key* in Figure 3.1(c)).

In this chapter, a hierarchical direct mapping algorithm is provided. The algorithm preserves the semantics with strict logical rules. The mapping problems occur when a mapping method simply focuses on a data type transformation. Thus, the method considers the prevention of such problem using a hierarchical semantic vocabulary and advanced mapping rules to process mapping without semantic information loss. An evaluation metric is also defined using inverse-mapping phase; a mapping method is assessed that it is semantics preserving if the result of inverse-mapping is semantically identical to the original input data (Figure 3.2).

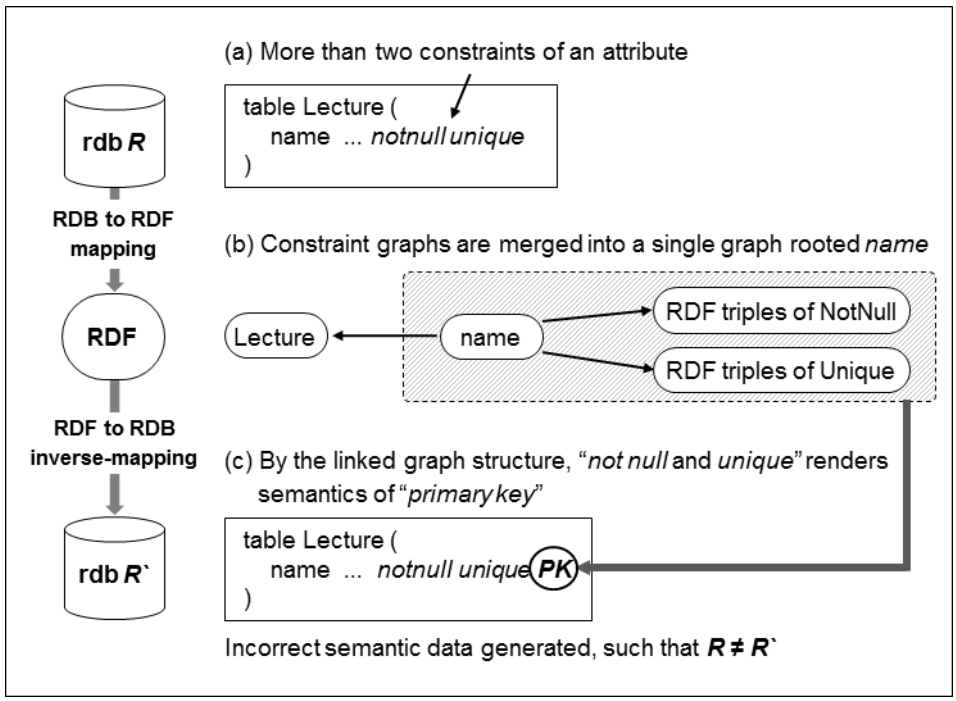



**Figure 3.1.** An example of problems during a direct mapping process.

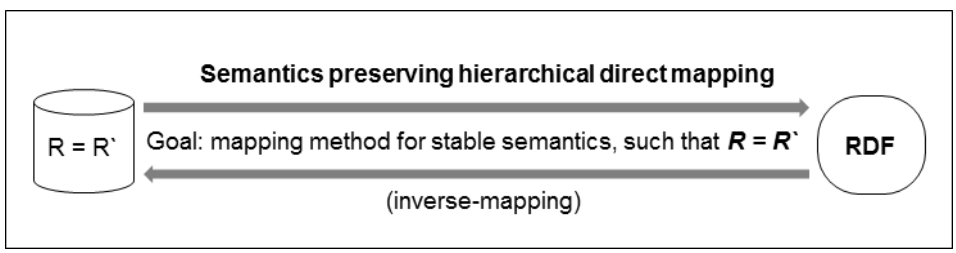



**Figure 3.2.** Overview of our direct mapping method.

# 3.2 Base Definitions of Predicates

The predicates in Table 3.1 are used for the verification of OWL ontology and to be used by the mapping rules in the next sections.

**Table 3.1.** List of predicates used in mapping rules.

| Predicates | Conditions of predicates to return true |
|---|---|
| Class(*r*) | *r* is an OWL class |
| Prop(*p, d, r*) | *p* is an RDF property with domain *d* and range *r* |
| ObjProp(*p, d, r*) | *p* is an OWL object property with domain *d* and range *r* |
| DataProp(*p, d, t*) | *p* is and OWL datatype property with domain *d* and datatype *t* |
| FP(*p*) | *p* is an OWL functional property |
| IFP(*p*) | *p* is an OWL inverse functional property |
| Card(*p, v*) | cardinality of property *p* is *v* |
| MinCard(*p, v*) | minimum cardinality of property *p* is *v* |
| MaxCard(*p, v*) | maximum cardinality of property *p* is *v* |
| type(*x, t*) | datatype of *x* is *t* |
| subClassOf(*x, y*) | *x* is a subclass of *y* |
| Rel(*r*) | *r* is a relation |
| BinRel(*r*, $a_{1,\ldots,m}$, *s*, $b_{1,\ldots,n}$, *t*) | *r* is a binary relation between relation *s* with primary key columns $a_{1,\ldots,m}$ and *t* with primary key columns $b_{1,\ldots,m}$ |
| Attr(*a, r*) | *a* is an attribute of relation *r* |
| FKeyAttr(*a, r*) | *a* is a foreign key |
| NonFKeyAttr(*a, r*) | *a* is not a foreign key |

# 3.3 Semantics Preservation

Semantics preservation is an important feature of the RDB to RDF mapping as the quality of direct mapping is majorly depends on the semantics preservation. Authors of [9, 10] provided a theoretical definition of semantics preservation. In addition to the definition, a more strict definition of semantics preservation is provided in this thesis. The provided definition is to quantify semantics preservation and to evaluate the accuracy of the mapping methods.

Suppose $X$ is a set of relational data, F is a RDB to RDF mapping function, and G is an RDF to RDB inverse-mapping function of F, then the semantics preservation of mapping methods is defined as follows:

**Semantics preservation:** For input data $X$, if $|X| = |G(F(X))|$ and $|G(F(X)) - X| = 0$, then F is an ideal function that satisfies semantics preserving mappings (Figure 3.3(a)).

**Loss of semantics:** For input data X, supposed that $|X| > |G(F(X))|$ and $X \supset G(F(X))$, then F loses semantics of relational data. The number of correctly transformed data by F is $|X \cap G(F(X))|$. The number of data lost by F is $|X| - |X \cap G(F(X))|$ (Figure 3.3(b) and (d)).

**Incorrect semantic data generation:** For input data $X$, supposed that $|X| < |G(F(X))|$ and $X \subset G(F(X))$, then there exists an incorrect semantic data generation in F. The number of incorrect semantic data generated by F is $|G(F(X)) - X|$ (Figure 3.3(c) and (d)).

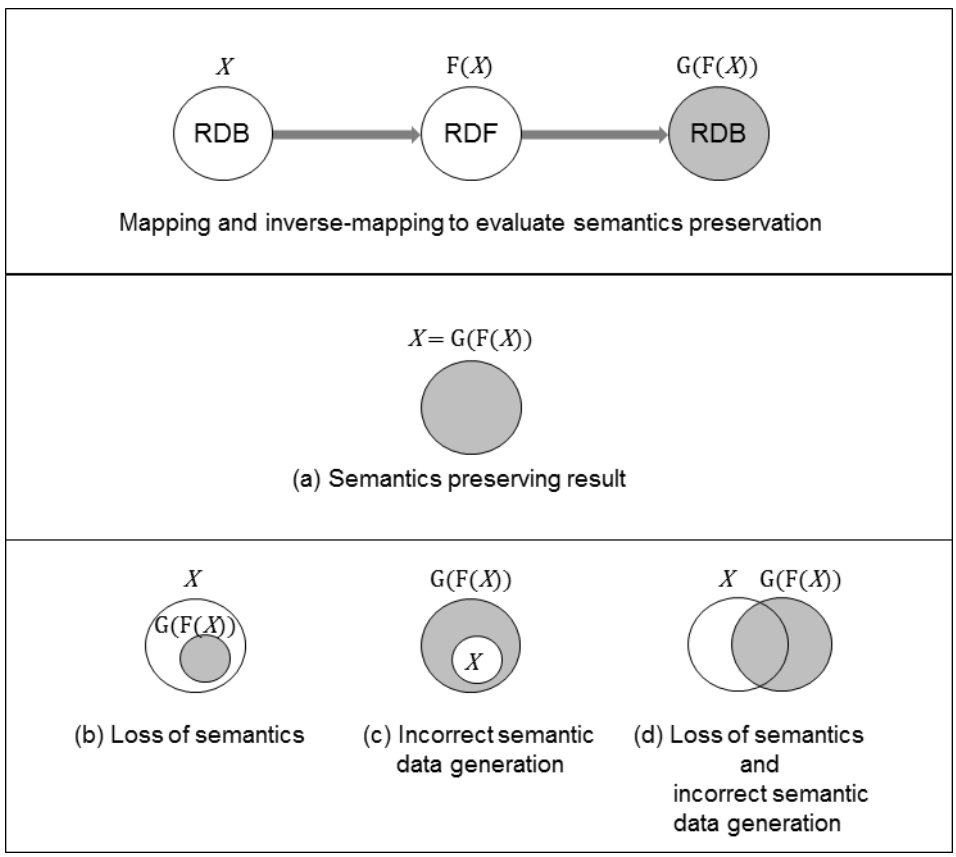



**Figure 3.3.** Semantics preservation of direct mapping.

# 3.4 Problem Description

In this section, three challenging problems, which may occur during direct mapping processes, are defined. The first problem and the second problem describe the loss of semantics, and the third problem describes the incorrect semantic data generation. These problems have been observed in specific cases. The problems and specific conditions in which the problems occurred were found during the study of existing direct mapping methods [1] [59, 62-67]. Consequently, the problems are organized in three categories to overcome these drawbacks in the next section.

**Problem 1:** Suppose $y_a$ = Class($x_a$) is an RDB to RDF mapping rule for a relational table, where $x_a \in R$, $R$ is a set of relational tables, $y_a \in C$, and $C$ is a set of OWL classes, $x_b$ = Class_Inverse($y_b$) is an RDF to RDB inverse-mapping rule of an OWL class, where $x_b \in X$, $X$ is a set of results generated by Class_Inverse($y_b$), $y_b \in C$, and $C$ is a set of OWL classes. However, $x_b$ is not the same as $x_a$ because $R \subset X$.

**Problem 2:** Suppose $y_a$ = ObjProp($x_a$) is an RDB to RDF mapping rule for a relational table, where $x_a = \{x_{a1}, x_{a2}\}$, $x_{a1} \in B$, $B$ is a set of binary

[1] Among mapping approaches, we especially analyzed [67] as this method improved previous approaches as well as provided practical implementation and experimental results.

relations, $x_{a2} \in F$, $F$ is a set of foreign keys, $y_a \in O$, and $O$ is a set of OWL object properties, $x_b$ = ObjProp_Inverse($y_b$) is an RDF to RDB inverse-mapping rule of an OWL object property, where $x_b \in X$, $X$ is a set of results generated by ObjProp_Inverse($y_b$), $y_b \in O$, and $O$ is a set of OWL object properties. However, ObjProp_Inverse( ) does not work as it was intended because ObjProp_Inverse( ) has not been given any information to determine whether $y_b$ is generated from $x_{a1}$ or $x_{a2}$.

**Problem 3:** Assume that mapping rules for integrity constraints of relational data are described in Figure 3.4. Here the predicates on the right hand side are used for verifying the integrity constraints. DefaultCondition($p$, $v$) is a function that assigns $v$ as a default value of predicate $p$. CheckCondition ($p$, $c$) is a function that assigns $c$ as a check condition of predicate $p$, and the other predicates on the left hand side are defined in section 3.2.

$$FP(p) \wedge Card(p,1) \rightarrow NotNull(p)$$
$$FP(p) \wedge IFP(p) \wedge MinCard\ (p,0) \wedge MaxCard\ (p,1) \rightarrow Unique(p)$$
$$FP(p) \wedge IFP(p) \wedge Card(p,1) \rightarrow PK(p)$$
$$FP(p) \wedge MinCard(p,0) \wedge MaxCard\ (p,1) \rightarrow FK(p)$$
$$FP(p) \wedge MinCard(p,0) \wedge MaxCard\ (p,1) \wedge DefaultCondition(p, v) \rightarrow Default(p, v)$$
$$FP(p) \wedge MinCard(p,0) \wedge MaxCard\ (p,1) \wedge CheckCondition(p, c) \rightarrow Check(p, c)$$

**Figure 3.4.** Simple mapping rules of integrity constraints.

Then we can infer the subset relationships as in Figure 3.5.

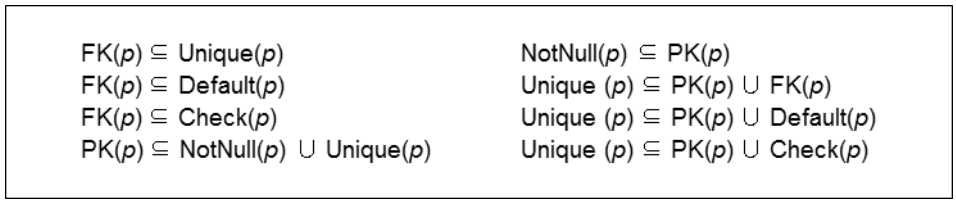



**Figure 3.5.** Subset relationships inferred by mapping rules of integrity constraints.

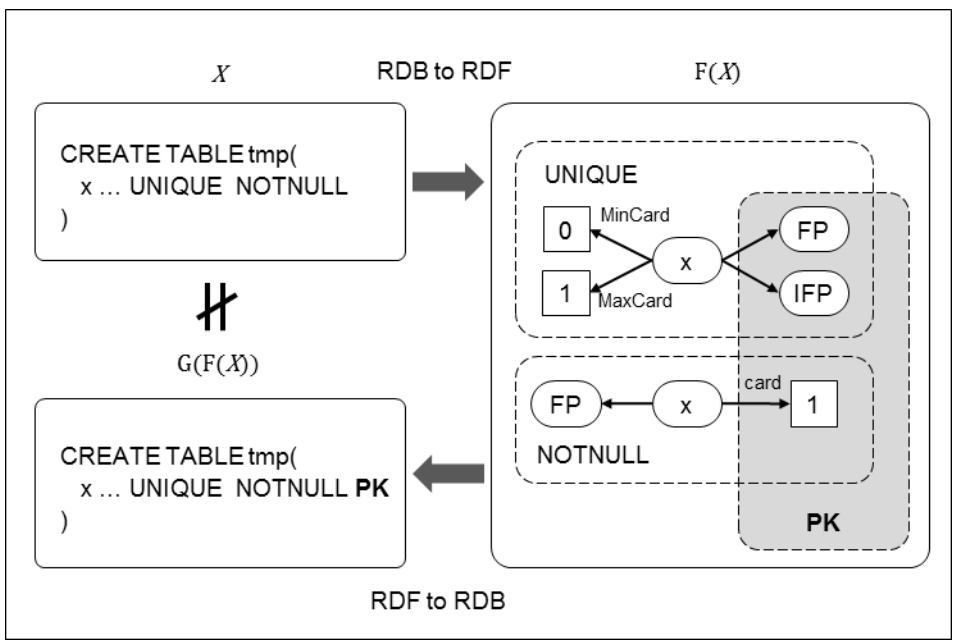



**Figure 3.6.** An example of Problem 3.

However, semantic data generated by the above rules can be misinterpreted. Figure 3.6 shows an example of Problem 3, assume that a relational attribute *x* has integrity constraints ‘unique’ and ‘not null’, F is a mapping function that contains the rules in Figure 3.4, and G is an inverse-mapping function of F, then the integrity constraints of G(F(*x*)) are ‘unique’, ‘not null’, and ‘primary key’ because FK(*p*) $\subseteq$ Unique(*p*), NotNull(*p*) $\subseteq$ Unique(*p*), and PK(*p*) $\subseteq$ NotNull(*p*) $\cup$ Unique(*p*).

# 3.5 Mapping Rules

In this section, mapping rules for learning general relational schemas and integrity constraints are provided. Each rule is based on lemmas, which are valid within the semantics domain. This section explains that the problems described in section 3.4 can be avoided by the rules using lemmas. The rules are defined using predicate logic, and graphical examples are provided for better understanding. A hierarchical structured semantic vocabulary is also provided to generate sound and precise semantic data.

### 3.5.1 Mapping Rules for General Relational Schemas

In this section, five rules are provided to generate accurate semantic data from general relational data. The rules are defined by utilizing Lemma 1 and Lemma 2 to avoid information loss (proofs are provided in the appendix).

Lemma 1 describes the feature of the OWL class during a mapping process.

**Lemma 1:** Suppose $R$ is a relational table set, $A$ is an attribute set, $K$ is an integrity constraint set, $I$ is a relational instance set, $X$ is a set where $X \subset (R \cup A \cup K \cup I)$, F is a direct mapping function. Then, we can retrieve every $y \in F(X)$ by inference from owl:Class.

Thus, to avoid Problem 1 described in section 3.4, we define Rule 1 for mapping relational tables based on Lemma 1 as:

**Rule 1:** $\text{Rel}(r) \wedge \neg\text{BinRel}(r, a1 \ldots m, s, b1 \ldots n, t) \rightarrow \text{Relation}(r)$,

where the predicates used on the left hand side are defined in section 3.2, and Relation($r$) is a predicate that verifies if $r$ is a relational table and not a binary relation. By Rule 1, relational tables are transformed to semantic resources using *Relation* (typed OWL class), which is a semantic vocabulary to notate relational tables.

For example, if a relational table *Student* is transformed by a naïve rule "Rel(*Student*)→Class(*Student*)", then the transformed output *Student* loses explicit information that it is a relational table. This loss happens because all semantic resources are typed only by OWL class (Figure 3.7(a)). On the other hand, *Student* will not lose its information that it is a relational table by the Rule 1. *Relation* is defined as a type of *Student* using Rule 1, which provides explicit information that the transformed semantic data is a relational table (Figure 3.7(b)).

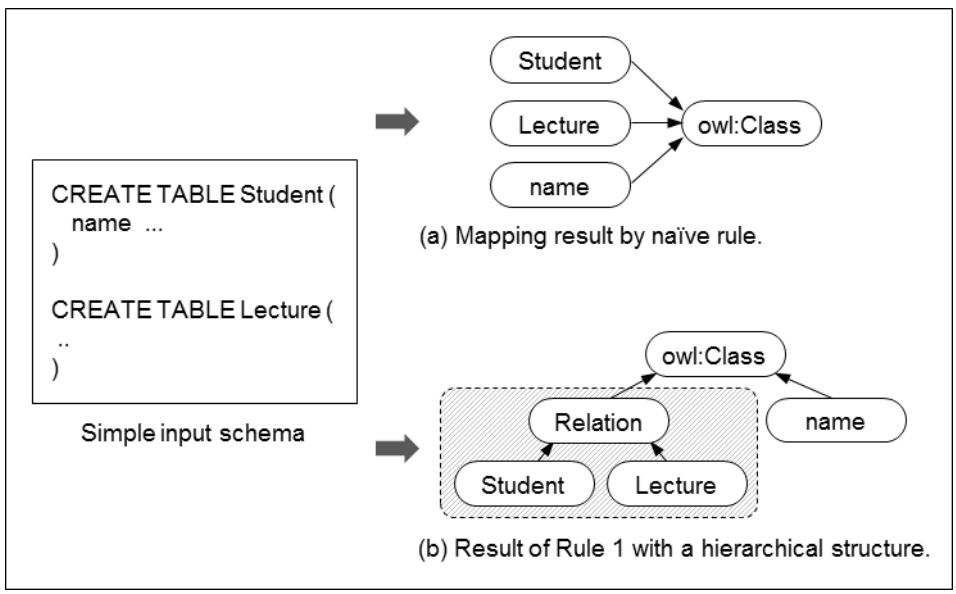



**Figure 3.7.** [Rule 1] A comparative example of rules for mapping relational tables.

Lemma 2 illustrates the feature of the OWL object property that is used to express the semantics of relationships between semantic resources.

**Lemma 2:** For any $X$ in relational data, if $x \in X$ references another $y \in X$, then $x$ can be transformed into a semantic resource, which has a type of owl:ObjectProperty.

Based on Lemma 2, if a direct mapping method does not manage the feature of an object property accurately, then Problem 2 described in section 3.4 can occur during the mapping process. Thus, Rule 2 to Rule 5 are defined based on Lemma 2 for mapping semantics of relationships.

Rule 2 specifies the attributes within the hierarchical structure:

**Rule 2:** $\text{Prop}(a, r, \_) \wedge \text{FP}(a) \rightarrow \text{Attr}(a, r)$

$\text{ObjProp}(a, r, s) \rightarrow \text{FKeyAttr}(a, r, s)$

$\text{DataProp}(a, r, \text{type}(a)) \rightarrow \text{NonFKeyAttr}(a, r)$

$\forall a \forall r\ \text{FKeyAttr}(a, r, s) \subseteq \forall a \forall r\ \text{Attr}(a, r)$

$\forall a \forall r\ \text{NonFKeyAttr}(a, r) \subseteq \forall a \forall r\ \text{Attr}(a, r),$

where the predicates on the left hand side are defined in section 3.2, and the predicates on the right hand side represent transforms of a relational attribute $a$. Rule 2 with these predicates has a distinct advantage over previous approaches. Previous approaches simply use OWL object property and datatype property to map relational attributes. Since the OWL properties are provided to describe any resources with referencing semantics not just for relational attributes, using only

the OWL properties does not always guarantee that output data was originally attribute data. As a result, previous approaches cannot avoid semantic information loss during mapping attributes. On the other hand, Rule 2 adopts hierarchical structured semantic vocabularies on attributes (Figure 3.8). The vocabularies are able to describe various types of attributes, and every input attributes can be transformed into semantic data with detailed information.

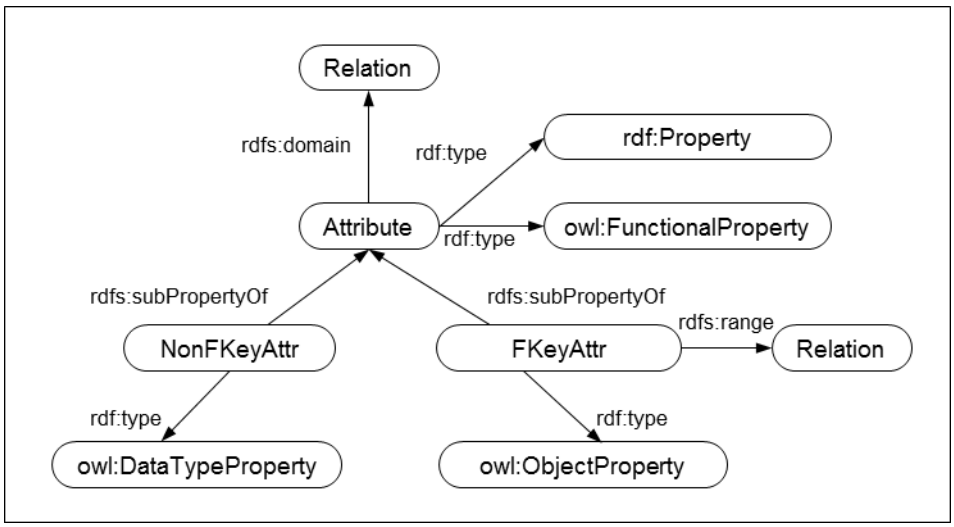



**Figure 3.8.** [Rule 2] Set of attributes as an hierarchical structured semantic vocabulary.

Rule 3 is for mapping binary relations, as follows:

**Rule 3:** BinRel(*r, a1…m, s, b1…n, t*) ∧ ¬BinRel(*s*, _, _, _, _) ∧ ¬BinRel(*t*, _, _, _, _) → BinaryRelation(*r*, *s*, *t*),

where the predicates on the left hand side are defined in section 3.2, and BinaryRelation(*r, s, t*) is a predicate that verifies if a binary relation *r* can be transformed into semantic resource *BinaryRelation* (typed OWL object property), which is a semantic vocabulary to notate binary relations. Even though both Rule 2 and Rule 3 use owl:ObjectProperty during a mapping process, the mapping results of the two rules can be distinguished individually. The semantics of type owl:ObjectProperty is encapsulated by the mapping resource *FKeyAttr* in Rule 2 (Figure 3.8) and *BinaryRelation* in Rule 3 (Figure 3.9). Therefore, a mapping result with *FKeyAttr* implies that it was originally an attribute of relational data, and a result having *BinaryRelation* can also be inferred that it was a binary relation before the mapping process.

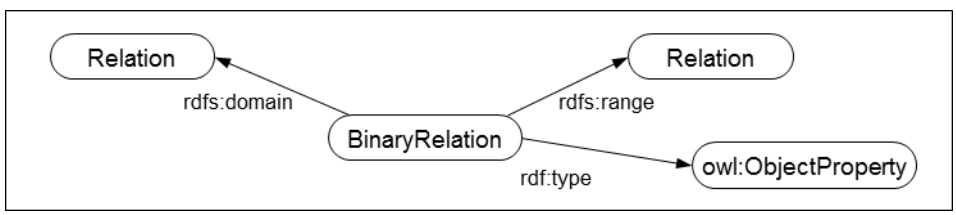



**Figure 3.9.** [Rule 3] Semantic vocabulary of a binary relation.

Rule 4 and Rule 5 indicate the relationships between relational tables:

**Rule 4:** Rel(*s*) ∧ Rel(*t*) ∧ PK(*a, s*) ∧ FK(*a, s, _, t*) ∧ ObjProp(*r, s, t*) ∧ FP(*r*) → IdentifyingRelationship(*r, s, t*)

**Rule 5:** Rel($s$) ∧ Rel($t$) ∧ PK($a$, $s$) ∧ ¬FK($a$, $s$,_,$t$) ∧ ObjProp($r$, $s$, $t$) ∧ FP($r$) → NonIdentifyingRelationship($r$, $s$, $t$)

where the predicates on the left hand side are defined in section 3.2, IdentifyingRelationship($r$, $s$, $t$) is a predicate that verifies identifying relationships, and NonIdentifyingRelationship($r$, $s$, $t$) is a predicate that verifies non-identifying relationships.

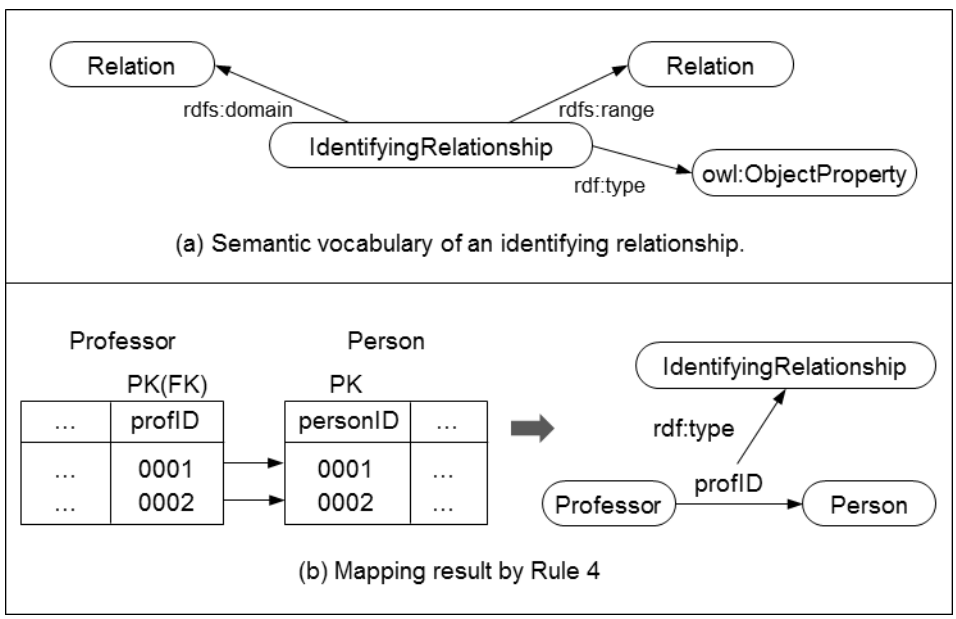



**Figure 3.10.** [Rule 4] A definition and an example of identifying relationship.

Figure 3.10 shows an example of mapping an identifying relationship. Since a primary key of *Professor* contains a foreign key referencing *Person*, relation *Professor* is dependent on relation *Person*. In such case, relationships between *Professor* and *Person* can be mapped using IdentifyingRelationship( ) as defined by Rule 4.

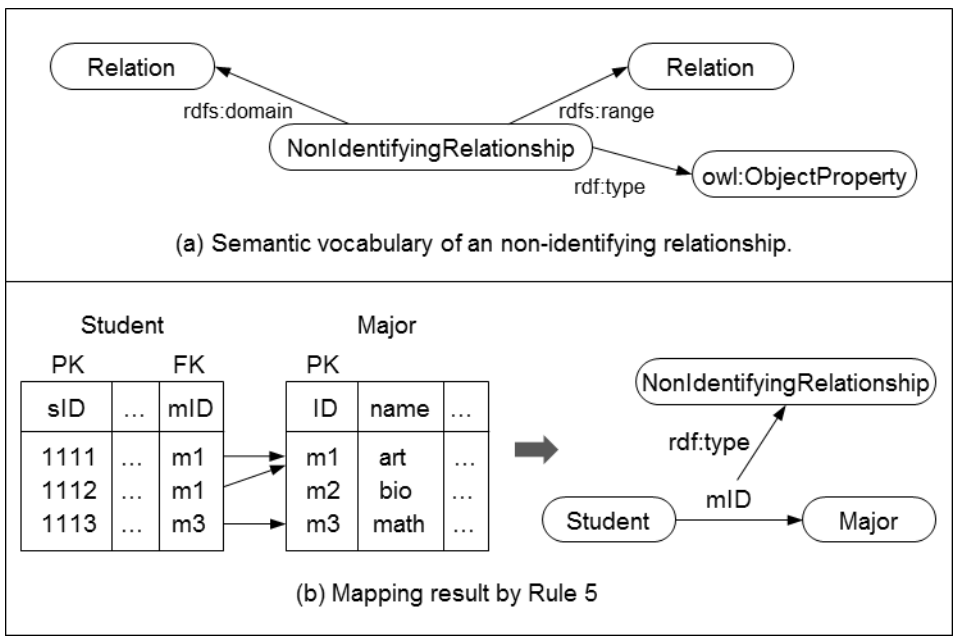



**Figure 3.11.** [Rule 5] A definition and an example of non-identifying relationship.

Figure 3.11 is an example of mapping a non-identifying relationship. Since *Student*'s foreign key referencing *Major* is not an attribute for the primary key of *Student*, relation *Student* and *Major* are independent. In such case, relationships

between *Student* and *Major* can be mapped using NonIdentifyingRelationship( ) by Rule 5.

### 3.5.2 Mapping Rules for Integrity Constraints

In this section, six rules are provided to transform relational integrity constraints, and to prevent incorrect semantic data generation problems. Lemma 3 illustrates the feature of semantic data using a linked graph structure. This feature acts as the major factor to avoid generation of incorrect semantic data. Thus, Rule 6 through 11 are defined for mapping integrity constraints based on Lemma 3 (the proof is provided in the appendix).

**Lemma 3:** Suppose $G$ is an RDF graph, $G_1$ and $G_2$ are components of $G$, there is no edge between $G_1$ and $G_2$, $G_1$ is rooted at $x \in R$, $G_2$ is rooted at $y \in R$, where $R$ is a set of semantic resources, then,

(1) If $x$ and $y$ have the same URI, then $x$ is identical with $y$. Thus, $G_1$ and $G_2$ can be merged into one graph.

(2) If $x$ and $y$ have different URIs, $x$ has a property $p_1$, and $y$ has a property $p_2$ that has the same URI as $p_1$, then $G_1$ and $G_2$ cannot be merged into one graph, and $p_1$ can be distinguished from $p_2$ using $x$ and $y$.

**Rule 6:** NonFKeyAttr(*a*, *r*) ∧ subClassOf(*r*, *_b*) ∧ Card(*a*, *_b*, *1*) → NotNull(*a*, *r*)

**Rule 7:** NonFKeyAttr(*a*, *r*) ∧ IFP(*a*) ∧ subClassOf(*r*, *_b*) ∧ MaxCard(*a*, *_b*, *1*) ∧ (∃ !*v*) a(*r*, *v*) → Unique(*a*, *r*)

**Rule 8:** NonFKeyAttr(*a*, *r*) ∧ IFP(*a*) ∧ subClassOf(*r*, *_b*) ∧ Card(*a*, *_b*, *1*) ∧ (∃ !*v*) a(*r*, *v*) → PK(*a*, *r*)

**Rule 9:** FKeyAttr(*a*, *r*, *s*) ∧ subClassOf(*r*, *_b*) ∧ MinCard(*a*, *_b*, *1*) → FK(*a*, *r*, *s*)

**Rule 10:** NonFKeyAttr(*a*, *r*) ∧ subClassOf(*r*, *_b*) ∧ MaxCard(*a*, *_b*, *1*) ∧ DefVal(*a*, *_b*, *v*) → Default(*a*, *r*)

**Rule 11:** NonFKeyAttr(*a*, *r*) ∧ subClassOf(*r*, *_b*) ∧ MaxCard(*a*, *_b*, *1*) ∧ CheckCond(*a*, *_b*, *v*) → Check(*a*, *r*),

where *a* is an attribute of relational table *r*, *_b* is a blank node, *v* in Rule 7 and 8 is an attribute value, *v* in Rule 10 is a default attribute value, and *v* in Rule 11 is a check condition. The predicates on the left hand side are defined in section 3.2, and

the predicates on the right hand side preserve integrity constraints: not null, unique, primary key, foreign key, default, and check.

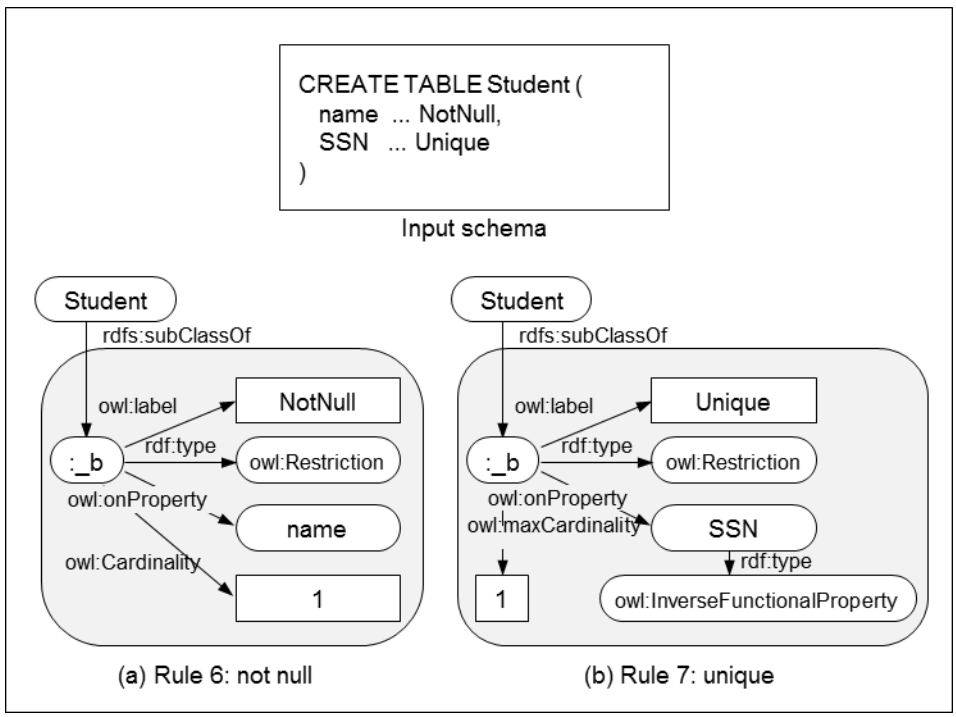



**Figure 3.12.** [Rule 6, 7] Examples of mapping integrity constraints.

Rule 6 describes not null constraint. It also defines a predicate Card(*a*, *_b*, *1*) that restricts the cardinality of an attribute to be exactly one (Figure 3.12(a)). Rule 7 specifies unique constraint, and defines a predicate with a unique existential quantifier ( $\exists$ !*v*) a(*r*, *v*) such that there is only one attribute value *v* contained in the domain of a(*r*, *v*) (Figure 3.12(b)).

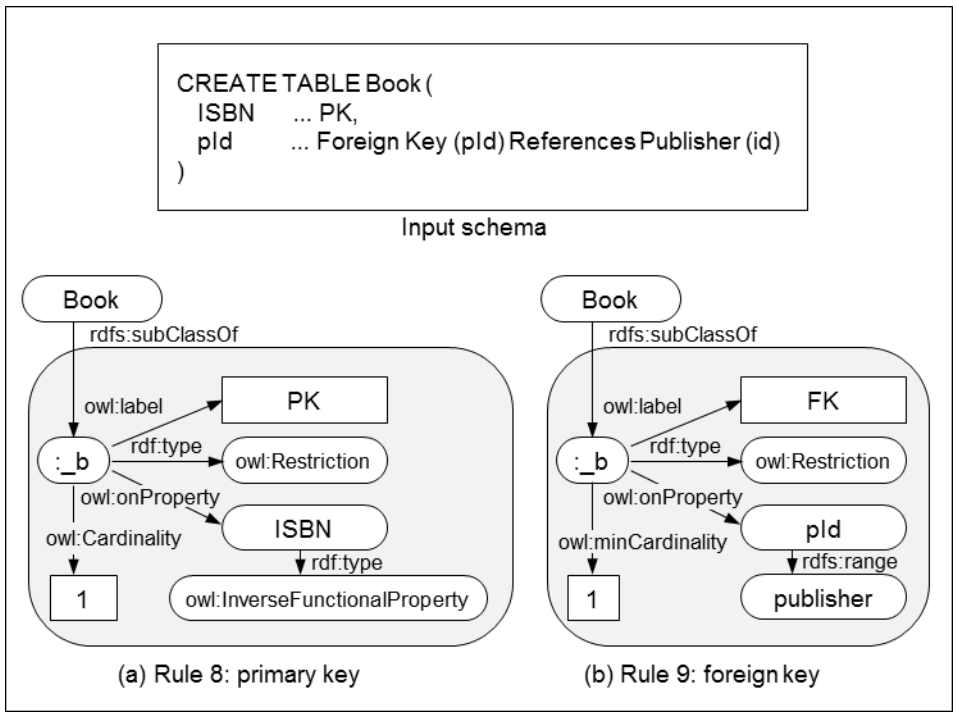



**Figure 3.13.** [Rule 8, 9] Examples of mapping integrity constraints.

Rule 8 specifies primary key defined by Card(*a*, *_b, 1*) and ($\exists !v$) a(*r*, *v*) to assign attribute *a* with a primary key (Figure 3.13(a)). To define a foreign key constraint, Rule 9 uses MinCard(*a*, *_b, 1*) to specify a lower bound of the cardinality because relational tables are able to reference more than one other tables. Rule 9 also uses FKeyAttr(*a*, *r*, *s*) to describe the semantics that the type of attribute *a* is OWL object property with domain *r* and range *s* (Figure 3.13(b)).

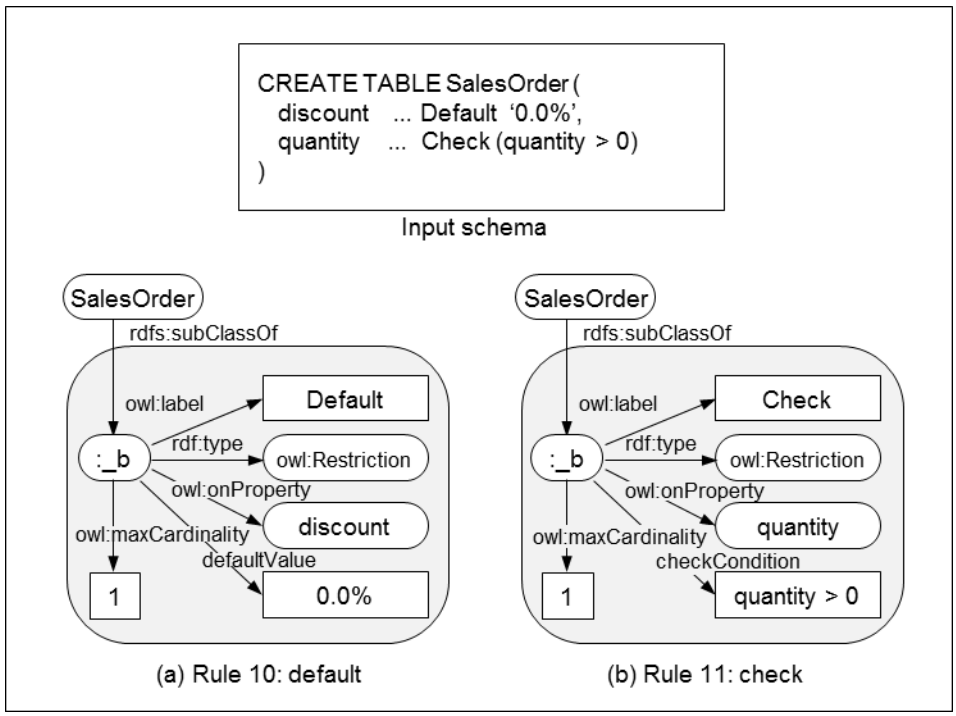



**Figure 3.14.** [Rule 10, 11] Examples of mapping integrity constraints.

Rule 10 specifies default constraint. Rule 10 uses a function DefVal(a, _b, v) that returns a default value v if a value of attribute a is omitted (Figure 3.14(a)). In Rule 11, a function CheckCond(a, _b, v) is used to restrict the value range for check constraint. For example, CheckCond(quantity, _b, 'quantity > 0') means that a value of the attribute quantity must be greater than zero (Figure 3.14(b)).

**Table 3.2.** Relationships among lemmas, rules, and problems.

| Lemmas | Rules | | Related data | Related problems |
|---|---|---|---|---|
| Lemma1 | General mapping rules | Rule 1 | Relational tables | Problem 1 |
| Lemma2 | | Rule 2<br>Rule 3<br>Rule 4<br>Rule 5 | Relational attributes<br>Binary relations<br>Identifying relationships<br>Non-identifying relationships | Problem 2 |
| Lemma3 | Rules for integrity constraints | Rule 6<br>Rule 7<br>Rule 8<br>Rule 9<br>Rule 10<br>Rule 11 | Integrity constraint: not null<br>Integrity constraint: unique<br>Integrity constraint: primary key<br>Integrity constraint: foreign key<br>Integrity constraint: default<br>Integrity constraint: check | Problem 3 |

### 3.5.3 Soundness and Completeness of the Rules

To show that provided mapping method is semantics preserving, the relationships among lemmas, rules, and problems are described in Table 3.2 for better understanding the concept of the rules. Lemmas describe the features of semantic resources during a mapping process. Lemma 1 states that every semantic resource can be inferred from OWL class (Semantic resource). Lemma 2 states that every semantic resource referencing other resources can be typed by OWL object property (Referential relationship). Lemma 3 states that every sub-graph, which has same semantic resource as a root node, can be merged into a single graph (Union of semantic resources). In this thesis, 11 relational elements for transformation are set based on the lemmas: relation, attribute, binary relation, identifying relationship,

non-identifying relationship, not null, unique, primary key, foreign key, default, and check. The elements are minimal set of input relational schema data for semantics preserving RDB to RDF transformation. On the other hand, problems described in section 3.4 are specific cases on violation of semantics preservation. First, Problem 1 illustrates the loss of information when relational tables are transformed without considering Lemma 1. Second, Problem 2 illustrates another loss of information when attributes, binary relations, or other referencing objects are transformed without considering Lemma 2. Finally, Problem 3 illustrates incorrect semantic data generation when integrity constraints are transformed without considering Lemma 3. Therefore, mapping rules are defined based on lemmas to perform semantics preserving RDB to RDF transformation and avoid the loss of semantics or incorrect semantic data generation by the problems.

The following lemma and theorem show soundness and completeness of the provided mapping rules in this thesis (proofs are provided in the appendix):

**Lemma 4:** Consider X is a set of relational data, F is an RDB to RDF mapping function, and G is an RDF to RDB inverse-mapping function of F. If mapping rules are defined based on Lemma 1, 2, and 3, then,

**(1) Soundness:** the mapping rules are sound that the rules generate only semantics in RDB data ($X \supseteq G(F(X))$).

**(2) Completeness:** the mapping rules are complete that the rules generate all semantics in RDB data ($X \subseteq G(F(X))$).

**Theorem 1:** The provided rules are sound and complete for the semantics preserving RDB to RDF transformation.

**Table 3.3.** Comparison of semantics preservation among general schema mapping.

| Input | M1 | M2 | M3 | M4 | M5 | M6 | M7 |
|---|---|---|---|---|---|---|---|
| Table | X | Δ | Δ | Δ | Δ | Δ | O |
| Attribute | Δ | Δ | Δ | Δ | Δ | Δ | O |

**Table 3.4.** Comparison of semantics preservation among relationships mapping.

| Input | M1 | M2 | M3 | M4 | M5 | M6 | M7 |
|---|---|---|---|---|---|---|---|
| Binary relation | X | O | Δ | X | O | O | O |
| Identifying relationship | Δ | Δ | Δ | Δ | Δ | Δ | O |
| Non-identifying relationship | X | X | X | X | Δ | Δ | O |

**Table 3.5.** Comparison of semantics preservation among constraints mapping.

| Input | M1 | M2 | M3 | M4 | M5 | M6 | M7 |
|---|---|---|---|---|---|---|---|
| Not null | Δ | Δ | Δ | Δ | Δ | Δ | O |
| Unique | X | Δ | Δ | Δ | Δ | Δ | O |
| Primary key | Δ | Δ | Δ | Δ | Δ | Δ | O |
| Foreign key | Δ | Δ | Δ | Δ | Δ | Δ | O |
| Default | X | X | X | X | X | Δ | O |
| Check | X | X | X | Δ | Δ | X | O |

Table 3.3, 3.4, and 3.5 show that comparison of semantics preserving among mapping methods. M7 is provided mapping rules in this thesis, and M1 through M5 are previous mapping rules: [90], [59], [62], [63], [64], and [67], respectively. Symbol "X" is used to denote that a mapping rule is not implemented, symbol "Δ" is used to denote that a mapping rule is implemented but not semantics preserving, and symbol "O" is used to denote that a mapping rule is implanted and semantics preserving. If a mapping rule is implemented based on the lemmas and theorem, then the rule performs semantics preserving mapping process. On the other hand, if a mapping rule does not consider the lemmas and theorem, then the rule cannot generate semantics preserving mapping result but occurs Problem 1, 2, or 3.

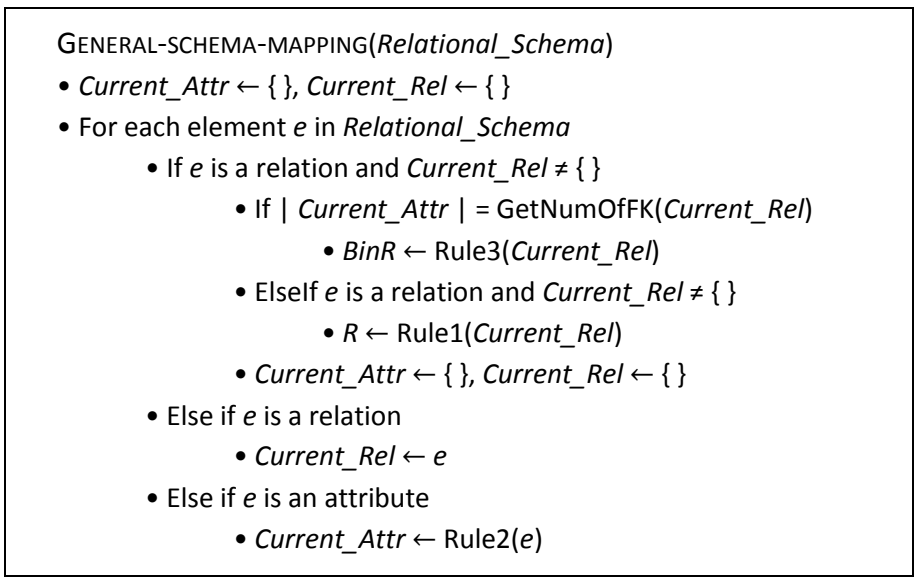



**Figure 3.15.** Pseudocode of the general schema mapping rules.

CONSTRAINT-SCHEMA-MAPPING(*Relational_Schema*)
- *Current_Attr* ← { }, *Current_Rel* ← { }
- For each element *e* in *Relational_Schema*
    - If *e* is a relation and *Current_Rel* ≠ { }
        - *Current_Attr* ← { }, *Current_Rel* ← { }
    - Else if *e* is a relation Then *Current_Rel* ← e
    - Else if *e* is an attribute Then *Current_Attr* ← e
    - Else if *e* is a constraint
        - *NN* ← Rule6IfExist(*Current_Rel, Current_Attr, e*)
        - UQ ← Rule7IfExist(*Current_Rel, Current_Attr, e*)
        - PK ← Rule8IfExist(*Current_Rel, Current_Attr, e*)
        - FK ← Rule9IfExist(*Current_Rel, Current_Attr, e*)
        - DF ← Rule10IfExist(*Current_Rel, Current_Attr, e*)
        - CK ← Rule11IfExist(*Current_Rel, Current_Attr, e*)
    - If e ∈ *FK* and e ∈ *PK* of *Current_Rel*
        - *IdR* ← Rule4(*Current_Rel, Current_Attr, e*)
    - Else if e ∈ *FK* and e ∉ *PK* of Current_Rel
        - *NIdR* ← Rule5(*Current_Rel, Current_Attr, e*)

**Figure 3.16.** Pseudocode of the constraint schema mapping rules.

## 3.5.4 Implementation

Figure 3.15 and 3.16 show algorithms of relational schema mapping rules. In these algorithms, *R* is a relation set generated by Rule 1. *A* is an attribute set generated by Rule 2. *BinR* is a binary relation set generated by Rule 3. *IdR* is an identifying relationship set generated by Rule 4. *NIdR* is a non-identifying relationship set generated by Rule 5. *NN* is a not null constraint set generated by Rule 6. *UQ* is a

unique constraint set generated by Rule 7. *PK* is a primary key constraint set generated by Rule 8. *FK* is a foreign key constraint set generated by Rule 9. *DF* is a default constraint set generated by Rule 10. *CK* is a check constraint set generated by Rule 11. Figure 3.15 is an algorithm for general schema mapping. The algorithm, which runs in $O(n^2)$ time, transforms relational input schema into semantic data by Rule 1, 2, and 3. Figure 3.15 is an algorithm, which runs in $O(n^2)$ time, for referential relationships by using Rule 4 and 5, and constraint schema mapping using Rule 6 through 11. The two algorithms are concurrently executable on a single machine by Lemma 3 (Union). Therefore, time complexity of processing provided mapping rules is $O(n^2)$.

The hierarchical direct mapping method is implemented on the Hadoop MapReduce framework [69, 70] to manage large-scale data on the Web. The MapReduce job framework of our mapping method is presented in Figure 3.17, and the pseudocode is provided in Figure 3.18. The algorithm receives relational data with schema information as an input data. Each map function generates semantic triples that contain schema information and writes the triples to the global cache file. The remaining instance data is emitted to reduce functions with keys using URIs of tables where the data is contained. Each reduce function generates semantic triples of relational instance data, which are processed using schema information in the global cache built in the map phase. Finally, reduce functions output semantic triples of relational data.

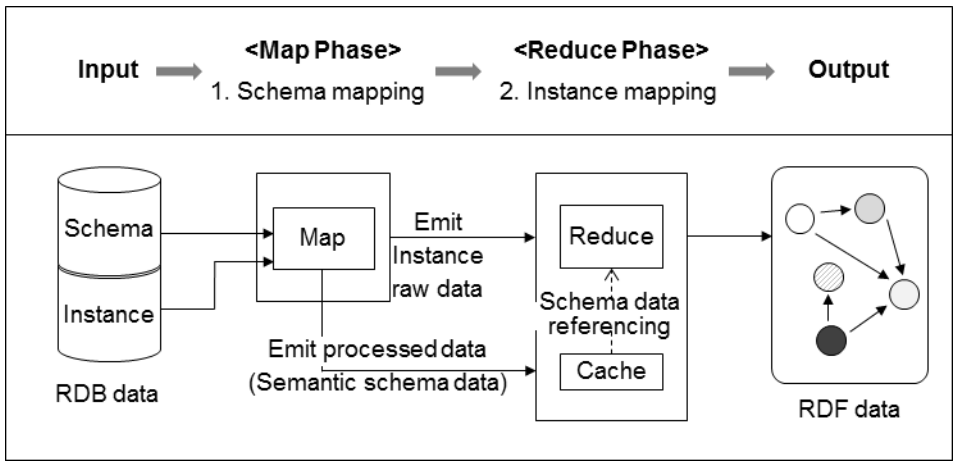



**Figure 3.17.** MapReduce job framework of hierarchical direct mapping.

```
class MAPPER
  method MAP(type t, element e)
    if isTable(t) then
      AddCache(getURI(e), buildTableSchema(e))              // Generate schema triples of a table
    else if isAttribute(t) then
      AddCache(getURI(getTable(e)), buildAttrSchema(e))     // Generate schema triples of an attribute
      if hasIntegrityConstraints(e) then                    // Generate integrity constraints
        AddCache(getURI(getTable(e)), buildIntegrityConstraints(e))
      end
    else                                  // Emit instance data
      EMIT(get(URI(getTable(e)), getValues(e))
    end

class REDUCER
  method REDUCE(table t, values [v1, v2, …])
    for all v ∈ values [v1, v2, …] do                       // Build instance data using schemas
      EMIT(t, buildTriples(t, getSchemaFromCache(t), v))
    end
```

**Figure 3.18.** Pseudocode of the implemented MapReduce algorithm.

# 3.6 Evaluation

## 3.6.1 Experimental Setup

Experiments were conducted on a cluster of twelve nodes with 3.1 GHz quad-core CPU, 4GB memory, and 2TB hard disk. We used Hadoop 1.2.1 for parallel processing. The mapping algorithms based on MapReduce were compiled by javac 1.6.0.

The experiments are conducted using five real datasets and one synthetic dataset. Each real dataset contains relational schema information with integrity constraints: Ensembl-compara (DB1), Ensembl (DB2), PHPmyadmin (DB3), and MusicBrainz (DB4) [71-74]. The DBT2 benchmark [75] was used for synthetic dataset. A warehouse data was generated using DBT2 and restructured the schema by adding integrity constraints to evaluate semantics preservation of the mapping methods. To perform a comparative analysis of semantics preserving direct mapping, the previous method was employed (OWL ontology based augmented direct mapping) [67], which provides implementation details of the mapping algorithm and shows improvement over previous methods.

## 3.6.2 Experimental Results

Figure 3.19 shows the results for the number of triples transformed from the

relational data. The horizontal axis represents relational data size as input of the mapping methods and the vertical axis is the number of semantic triples as an output data. From the Figure 3.19, the provided approach generates lesser number of triples compared to the previous method.

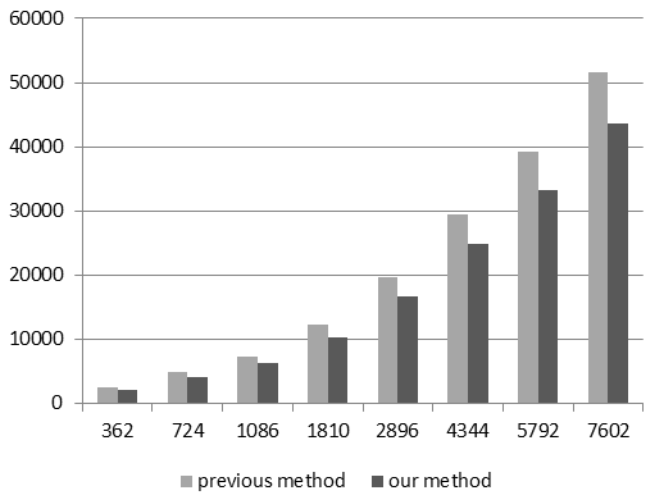



**Figure 3.19.** The number of output triples over relational data.

Figure 3.20 shows the average number of triples as a result of each transformation method. The horizontal axis represents each relational dataset and the vertical axis is the average number of triples generated from transforming a single relational element. Assuming that two output results are identical in semantics, the method that generates a smaller size result is better in both space and computation. Thus, the results show that the proposed approach generates more compact semantic data that expresses the same information with fewer resources.

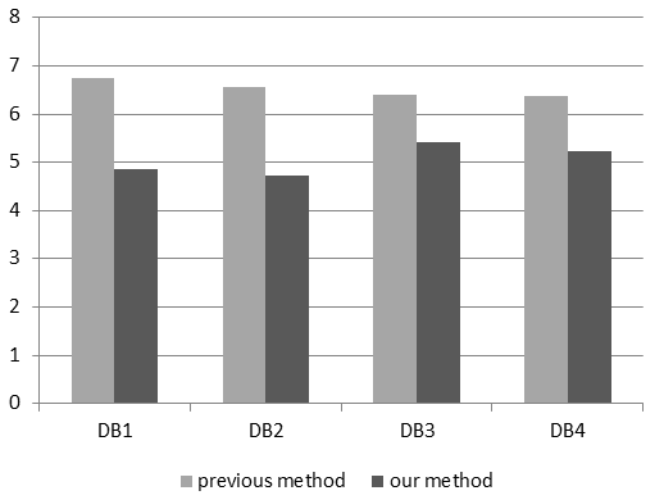



**Figure 3.20.** The average number of output triples for one input data.

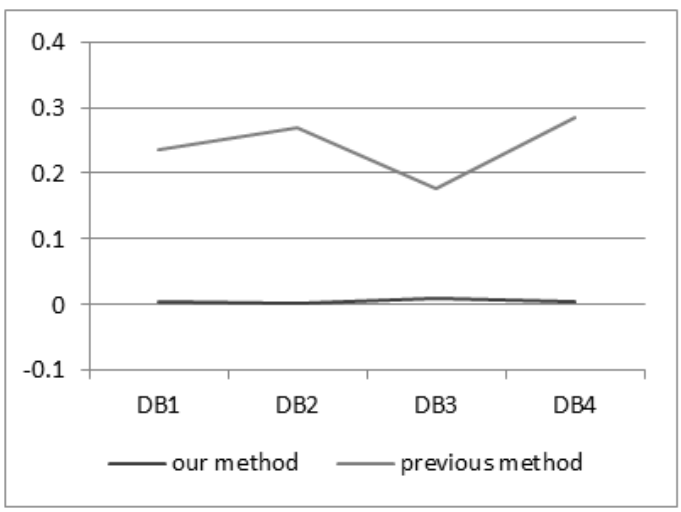



**Figure 3.21.** RDB2RDF failure rate of the mapping methods.

Figure 3.21 shows the failure rate of mapping methods in each database. The horizontal axis represents each relational dataset and the vertical axis is the failure rate during the transformation of relational data into semantic data. The mapping failures of the previous approach results in the incorrect semantic data generation problems discussed in section 3.4. These failures occur since the previous methods lack support in handling integrity constraints. On the other hand, the proposed approach improved the mapping rules to transform integrity constraints, and generate lesser false mapping results.

Figure 3.22 through 3.27 show completeness and soundness of mapping rules in each relational element. The horizontal axis represents each relational element: general schemas (Figure 3.22 and 3.25), schemas about referential relationships (Figure 3.23 and 3.26), and constraints (Figure 3.24 and 3.27). The vertical axis is recall (Figure 3.22, 3.23, and 3.24) or precision (Figure 3.25, 3.26, and 3.27). M1 through M4 are mapping methods for comparative analysis: [90], [64], [67], and the provided method in this chapter, respectively. The results show that the provided method follows the definition of the semantics preserving direct mapping rule, and provide sound and complete mapping results.

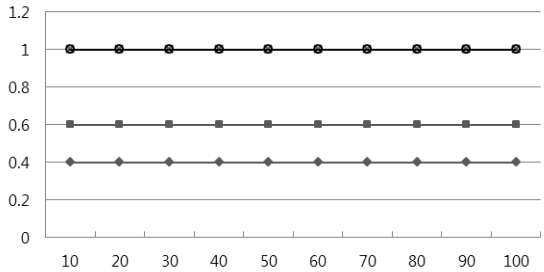



**Figure 3.22.** Completeness of mapping methods on general schemas.

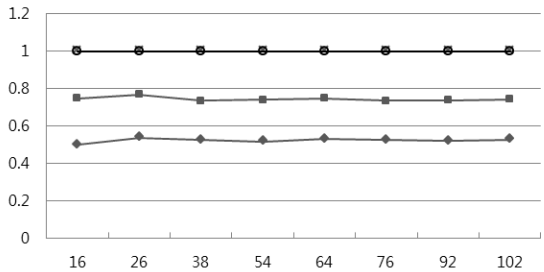



**Figure 3.23.** Completeness of mapping methods on referential relationships.

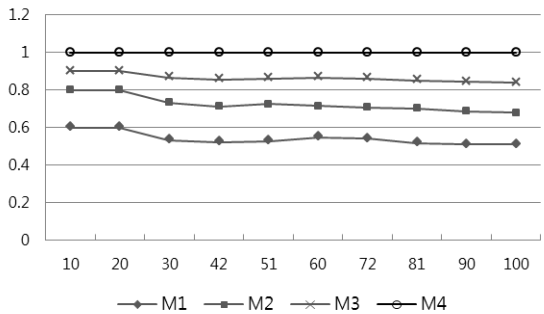



**Figure 3.24.** Completeness of mapping methods on constraints.

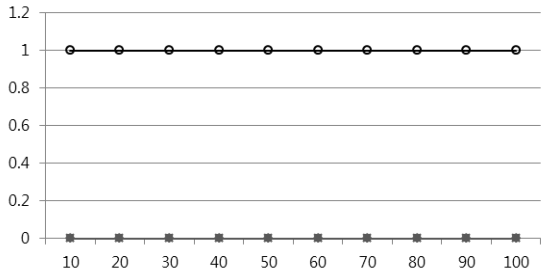



**Figure 3.25.** Soundness of mapping methods on general schemas.

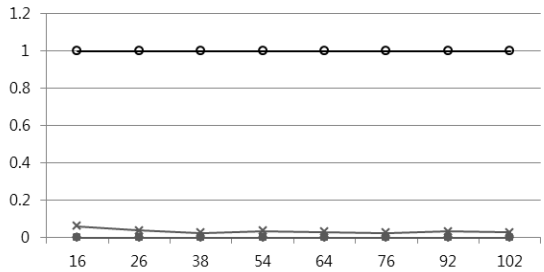



**Figure 3.26.** Soundness of mapping methods on referential relationships.

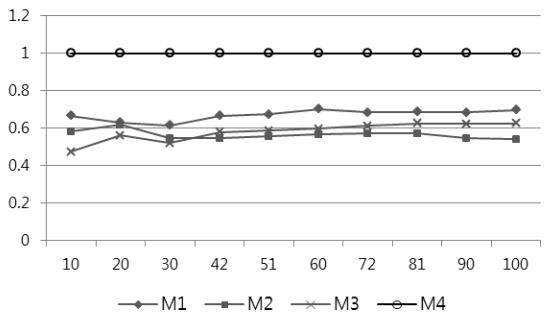



**Figure 3.27.** Soundness of mapping methods on constraints.

# Chapter 4
# Repetitive Data Reduction Methods for RDB to RDF Transformation

In this section, we present an overview of an optimized hierarchical direct mapping and advanced semantic vocabularies. We present a description of the optimized hierarchical direct mapping in three sections, focusing on respectively the multi-column key constraint, constraints of multiple keys, and metadata of integrity constraint. We also discuss the performance of the optimized hierarchical direct mapping in Section 4.6.

## 4.1 Motivation

Although semantics preserving RDB to RDF transformation is available using the

improved direct mapping, there is still a demand for developing efficient transformation methods for reducing output data size. The main reason of generating repetitive data is that attributes are able to be constrained by multiple integrity constraints.

The first case of generating repetitive data occurs when a key constraint as primary key, foreign key, or unique key comprises two or more attributes (Figure 4.1 as an example). Suppose a table has a primary key defined by a single attribute. In this case, the mapping result outputs a single RDF sub-graph representing a primary key constraint with the table and the attribute (Figure 4.1(a)).

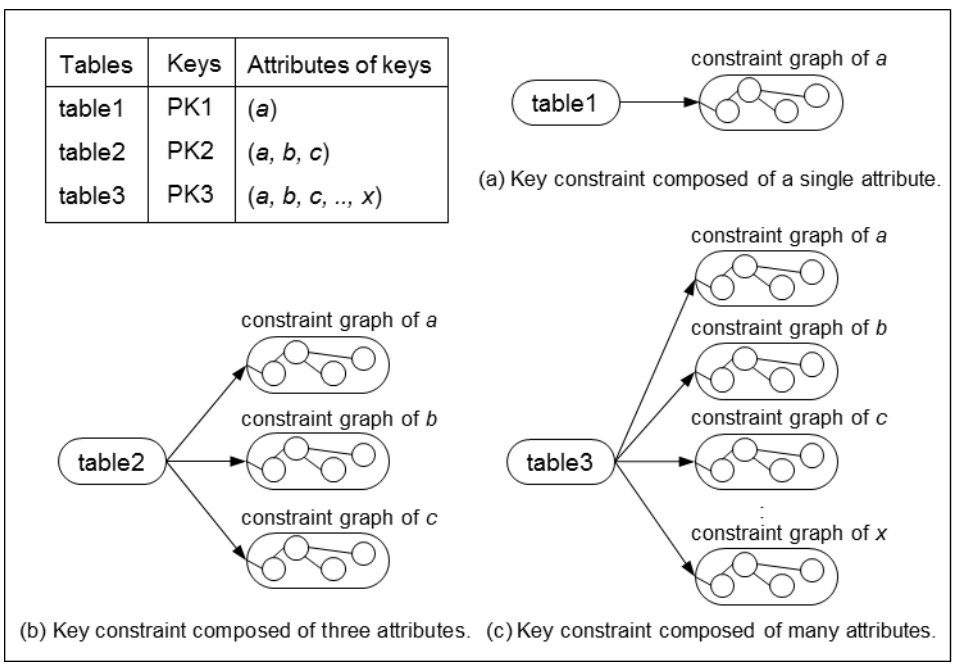



**Figure 4.1.** An example of generating repetitive semantic data in key constraints.

On the other hand, suppose a table has a primary key defined by three attributes. In this case, the mapping result outputs three RDF sub-graph to represent primary key constraint of the table with three attributes (Figure 4.1(b)). Moreover, if a table defines a primary key with several attributes, then the RDB to RDF mapping method outputs semantic data containing multiple primary key constraint RDF sub-graphs (Figure 4.1(c)). Semantic data containing multiple identical RDF sub-graphs does not corrupt the semantics preservation of RDB to RDF transformation, but repetitive data generation of mapping multi-column keys has a challenging issue of reducing repetitive semantic data without semantic information loss.

The second case of generating repetitive data occurs when a table contains two or more multi-column keys (Figure 4.2 as an example). Suppose $X$ is a dataset that contains a table, F is an RDB to RDF mapping function, and G is an RDF to RDB inverse-mapping function of F. If a table has two foreign keys defined by a single attribute respectively, then the result of F($X$) generates two foreign key constraint RDF sub-graphs and G(F($X$)) generates two foreign keys, such that $X$ = G(F($X$)) (Figure 4.2(a)). However, if a table has two foreign keys defined by two attributes respectively, then the result of F($X$) generates four foreign key constraint RDF sub-graphs and G(F($X$)) generates four foreign keys, such that $X \neq$ G(F($X$)) (Figure 4.2(b)). As the example shows that a semantic information loss occurs during RDB to RDF transformation of relational tables with two or more multi-column foreign keys, the second case has a significant demerit compared to the first case. Therefore, the second case has two challenging issues, reducing

repetitive semantic data and defining of semantics preserving mapping rules for relational tables that contain two or more multi-column foreign keys.

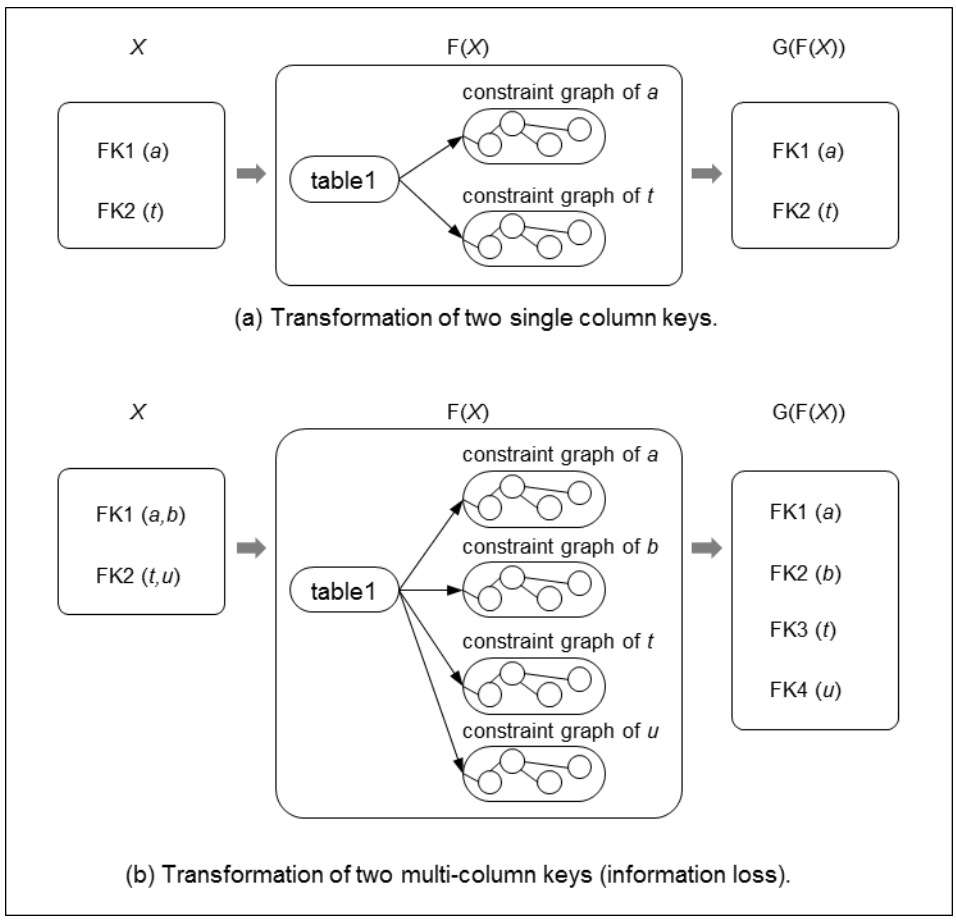



**Figure 4.2.** An example of information loss during mapping multiple key constraints.

The repetitive data generation problem occurs not only during mapping key constrains, but also mapping other integrity constraints (Figure 4.3). In the definition of relational database schema, each attribute is able to contain all integrity constraints. As a result, an RDB to RDF transformation of a relational schema that contains many integrity constraints output semantic data with several identical sub-graphs to express integrity constraints (Figure 4.3(a)). The transformed output does not corrupt the semantics preservation of RDB to RDF transformation as the first case explained above, but the repetitive data decreases readability with a complicate semantic data structure. Thus, the repetitive data generation of mapping relational data with many integrity constraints also has a challenging issue of reducing repetitive semantic data without semantic information loss (Figure 4.3(b)).

In this thesis, an optimized RDB to RDF transformation method is proposed based on a semantic relational meta-schema vocabulary. The proposed method transforms relational data into semantic data with lessor data size. The semantic relational meta-schema vocabulary defines relational tables, attributes, and integrity constraints in semantic notations. The vocabulary also defines relational resources with a more hierarchical structure to encapsulate complicate structures of constraint information and provides a lightweight and intuitive expression for mapping relational data.

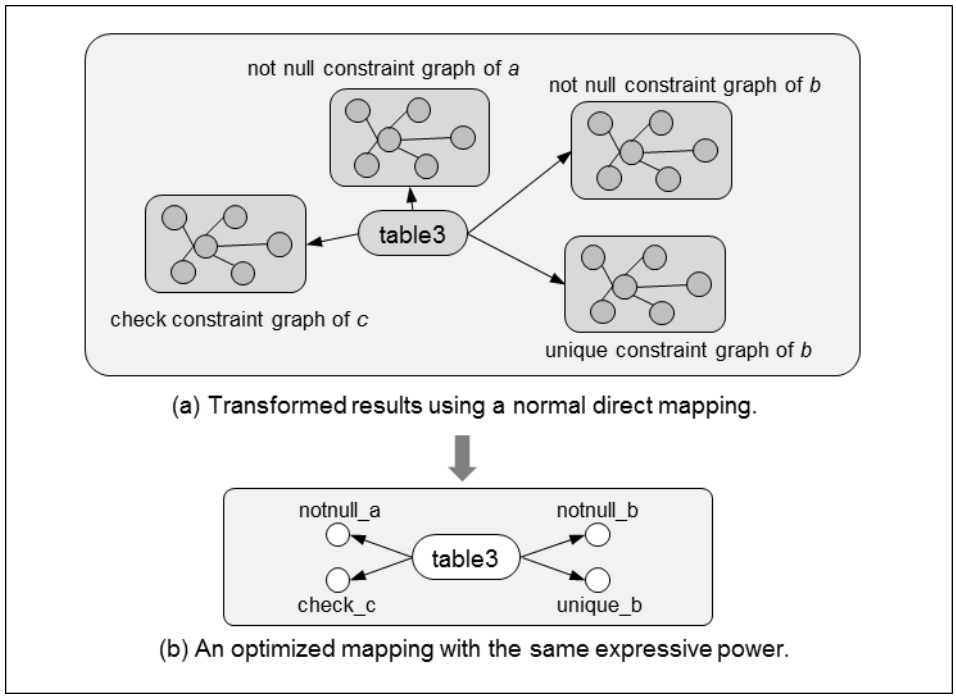



**Figure 4.3.** Optimized mapping that reduces repetitive semantic data.

# 4.2 Base Definitions of Predicates

The predicates in Table 4.1 are used for the verification of OWL ontology and to be used by the optimized mapping rules in the next sections.

**Table 4.1.** List of predicates used in optimized mapping rules.

| Predicates | Conditions of predicates to return true |
|---|---|
| Class(*r*) | *r* is an OWL class |
| Prop(*p, d, r*) | *p* is an RDF property with domain *d* and range *r* |
| Domain(*p, r*) | domain of *p* is *r* |
| Range(*p, s*) | range of *p* is *s* |
| ObjProp(*p, d, r*) | *p* is an OWL object property with domain *d* and range *r* |
| DataProp(*p, d, t*) | *p* is and OWL datatype property with domain *d* and datatype *t* |
| FP(*p*) | *p* is an OWL functional property |
| IFP(*p*) | *p* is an OWL inverse functional property |
| Card(*p, v*) | cardinality of property *p* is *v* |
| MinCard(*p, v*) | minimum cardinality of property *p* is *v* |
| MaxCard(*p, v*) | maximum cardinality of property *p* is *v* |
| type(*x, t*) | datatype of *x* is *t* |
| subClassOf(*x, y*) | *x* is a subclass of *y* |
| Rel(*r*) | *r* is a relation |
| BinRel(*r*, $a_{1,\ldots,m}$, *s*, $b_{1,\ldots,n}$, *t*) | *r* is a binary relation between relation *s* with primary key columns $a_{1,\ldots,m}$ and *t* with primary key columns $b_{1,\ldots,m}$ |
| Attr(*a, r*) | *a* is an attribute of relation *r* |
| NonFKeyAttr(*a, r*) | *a* is not a foreign key |

# 4.3 Mapping Multi-column Key

In this section, a set of rules are provided for mapping multi-column primary key, multi-column foreign key, and multi-column unique constraints. Each rule is based on lemmas defined in Chapter 3 to comply with semantics preservation. Predicate logic is used to define rules, and graphical examples are provided for better understanding.

### 4.3.1 Rules for Multi-column Primary Key

First, the following rule is for mapping primary key constraint that is comprised of a single attribute:

**Base primary key rule:** NonFKeyAttr($a$, $r$) ∧ IFP($a$) ∧ subClassOf($r$, $\_b$) ∧ Card($a$, $\_b$, $1$) ∧ ($\exists$ !$v$) a($r$, $v$) → PK($a$, $r$),

where $a$ is an attribute of relational table $r$, $\_b$ is a blank node, $v$ is an attribute value. The predicates on the left hand side are defined in section 4.2, and the rule specifies primary key defined by Card($a$, $\_b$, $1$) and ($\exists$ !$v$) a($r$, $v$) to assign attribute $a$ with a primary key.

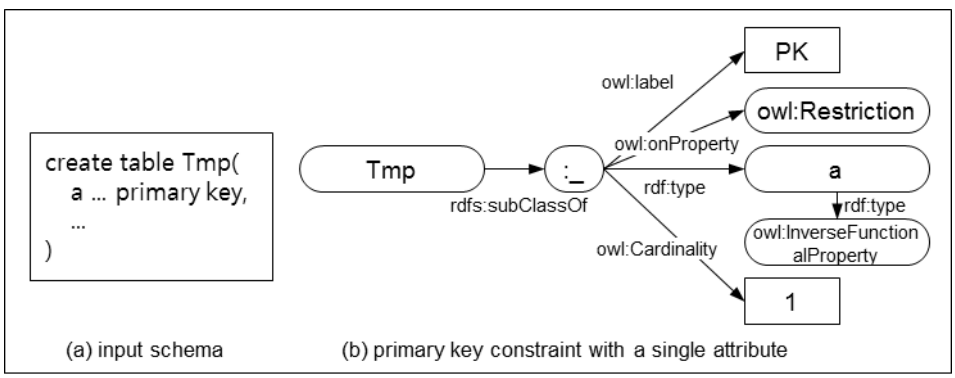



**Figure 4.4.** An example of mapping a single column primary key.

If the base primary key rule is used to transform a single column primary key, then a single sub-graph expressing a primary key constraint is generated by the mapping rule (Figure 4.4). On the other hand, if the base primary key rule is used to transform a table with a primary key that is comprised of three attributes, then three sub-graphs expressing the primary key constraint are generated by the mapping rule (Figure 4.5). The output result not only contains repetitive sub-graphs of constraint data, but can also be misinterpreted as the table has three primary keys.

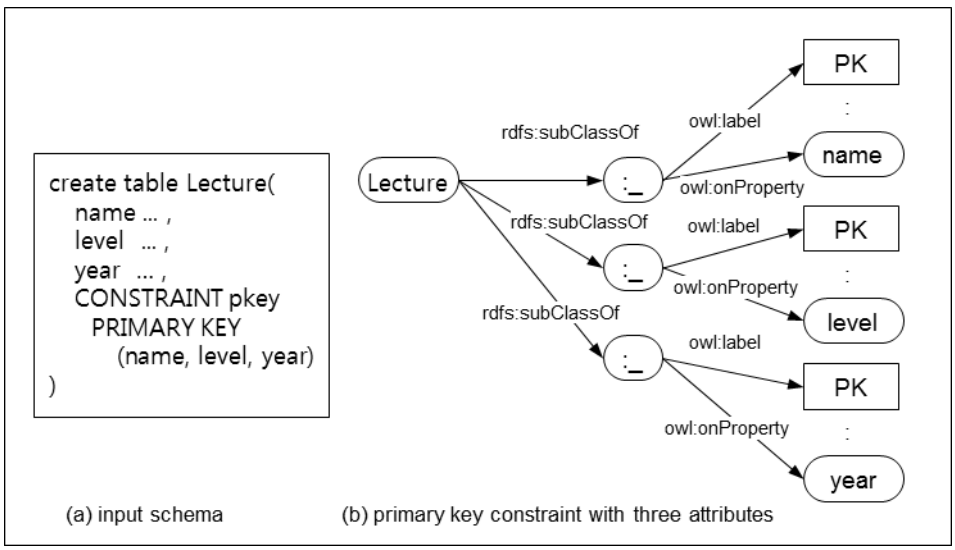



**Figure 4.5.** Transformation of a multi-column primary key using base primary key rule.

To avoid the misinterpretation problem, the base primary key rule is modified into grouped primary key rule as follows:

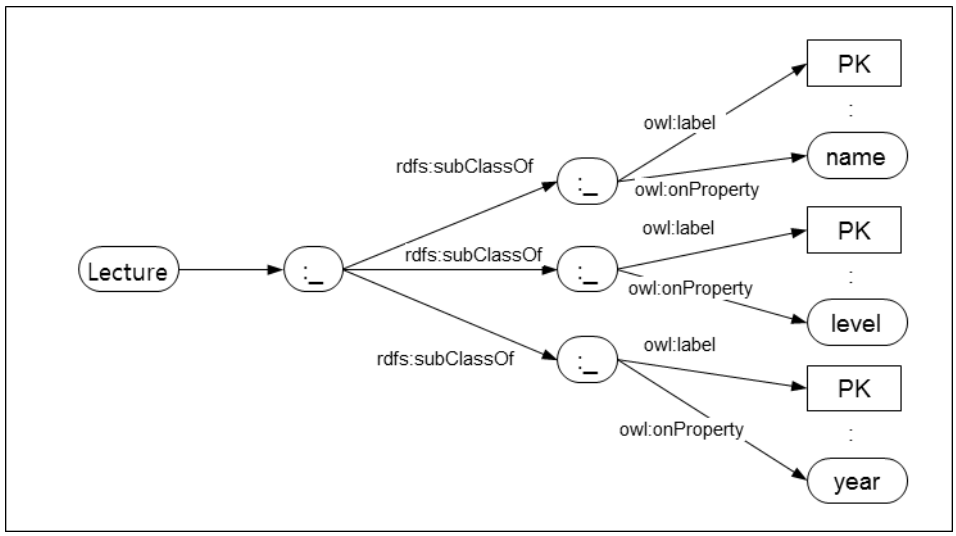



**Figure 4.6.** An example of transformation using grouped primary key rule.

**Grouped primary key rule:** NonFKeyAttr(*A*, *r*) ∧ IFP(*A*) ∧ subClassOf(*r*, *_g*) ∧ subClassOf(_g, *_B*) ∧ Card(*A*, *_B*, *1*) ∧ (∃!*V*) A(*r*, *V*) → PK(A, *r*),

where *A* is a set of attributes which a primary key is composed of, *r* is a relational table that contains *A*, *_B* is a blank node set of primary key constraints of *A*, *_g* is a blank node for grouping, *V* is a value set of *A*. The predicates on the left hand side are defined in section 4.2, and the rule specifies primary key defined by Card(*A*, *_B*, *1*) and (∃!*V*) A(*r*, *V*) to assign attribute set *A* with a primary key. The output result contains a merged primary key sub-graph so that the misinterpretation

problem can be avoided. However, the repetitive constraint data generation problem still exists (Figure 4.6).

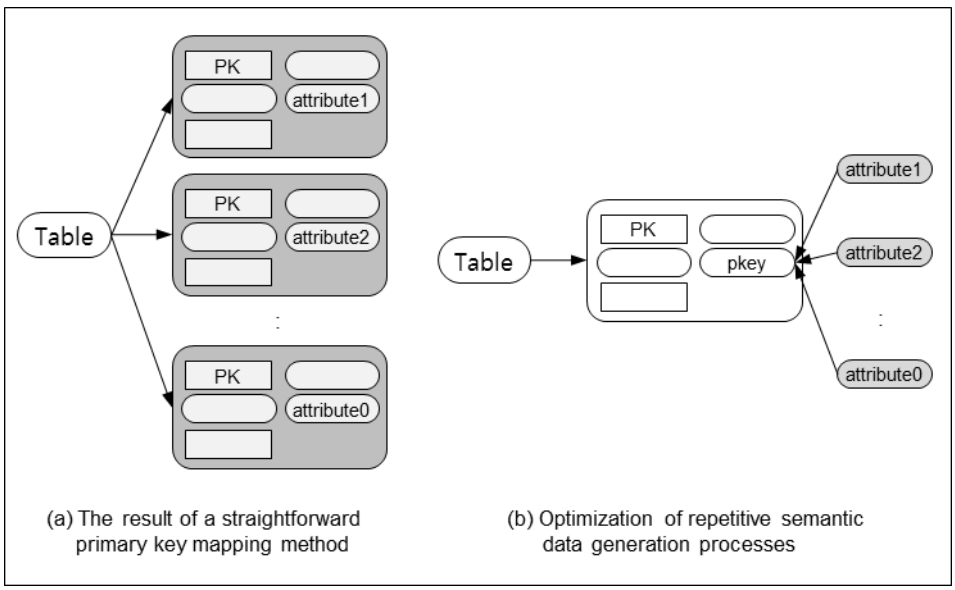



**Figure 4.7.** Transformation of a multi-column primary key using optimized primary key rule.

Thus, optimized the primary key rule is defined as follows:

**Optimized primary key rule:** NonFKeyAttr(*pkey*, *r*) ∧ IFP(*pkey*) ∧ subClassOf(*r*, *_b*) ∧ Card(*pkey*, *_b*, *1*) ∧ (∃!*V*)A(*r*, *V*) ∧ type(*A*, *pkey*) → PK(*A*, *r*),

where *A* is a set of attributes which a primary key is composed of, *r* is a relational table that contains *A*, *pkey* is an OWL property to represent a primary key constraint of *r*, *_b* is a blank node, *V* is a value set of *A*. The predicates on the left hand side are defined in section 4.2, and the rule specifies primary key defined by Card(*pkey*, *_b*, *1*), ($\exists !V$)A(*r*, *V*), and type(*A*, *pkey*) to assign attribute set *A* with the primary key. As shown in Figure 4.7, the optimized primary key rule outputs lessor constraint data. Instead of producing every primary key constraint sub-graphs (Figure 4.7(a)), the optimized primary key rule only generates a single primary key constraint, which is linked by primary key column attributes (Figure 4.7(b)). The output result contains a merged primary key sub-graph so that the misinterpretation problem can be avoided. Moreover, the rule generates compact output data with reducing repetitive constraint data.

## 4.3.2 Rules for Multi-column Foreign Key

The following rule is for RDB to RDF mapping of foreign key constraint defined by a single attribute:

**Base foreign key rule:** FKeyAttr (*a*, *r*, *s*) $\wedge$ subClassOf(*r*, *_b*) $\wedge$ MinCard(*a*, *_b*, *1*) $\rightarrow$ FK(*a*, *r*, *s*),

where *a* is an attribute of relational table *r*, *_b* is a blank node, *s* is a table referenced by table *r*. The predicates on the left hand side are defined in section 4.2,

and the rule uses MinCard(*a, _b, 1*) to specify a lower bound of the cardinality because relational tables are able to reference more than one other tables. Base foreign key rule also uses FKeyAttr(*a, r, s*) to describe the semantics that the type of attribute *a* is OWL object property with domain *r* and range *s*.

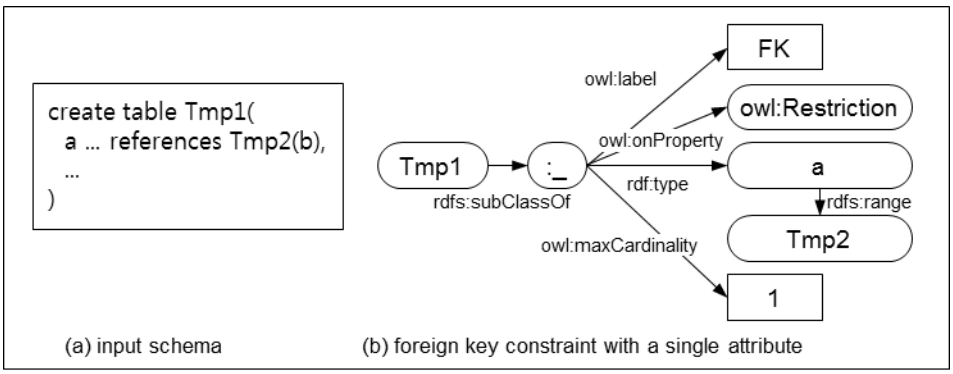



**Figure 4.8.** An example of mapping a single column foreign key.

If the base foreign key rule is used to transform a single column foreign key, then a single sub-graph expressing a foreign key constraint is generated by the mapping rule (Figure 4.8). On the other hand, if the base foreign key rule is used to transform a table with a foreign key that is comprised of three attributes, then three sub-graphs expressing the foreign key constraint are generated by the mapping rule (Figure 4.9). The output result not only contains repetitive sub-graphs of constraint data, but can also be misinterpreted as the table has three foreign keys.

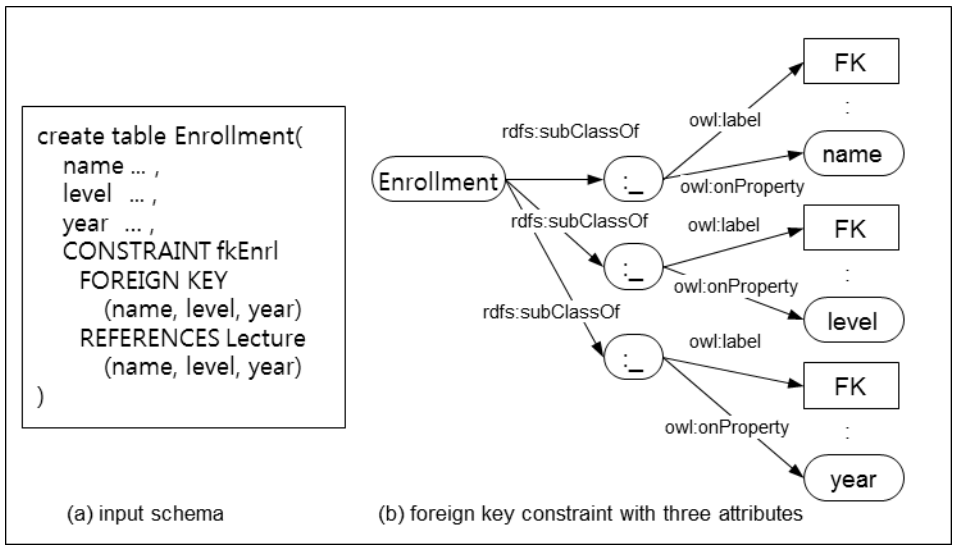



**Figure 4.9.** Transformation of a multi-column foreign key using base foreign key rule.

To avoid the misinterpretation problem, the base foreign key rule is modified into grouped foreign key rule as follows:

**Grouped foreign key rule:** FKeyAttr(*A*, *r*, *s*) ∧ subClassOf(*r*, *_g*) ∧ subClassOf(_g, *_B*) ∧ MinCard (*A*, *_B*, *1*) → FK(*A*, *r*, *s*),

where *A* is a set of attributes which a foreign key is composed of, *r* is a relational table that contains *A*, *s* is a table referenced by table *r*, *_B* is a blank node set of foreign key constraints of *A*, *_g* is a blank node for grouping. The predicates on the left hand side are defined in section 4.2, and the rule specifies foreign key defined by MinCard(*A*, *_B*, *1*) to assign attribute set *A* with a foreign key. The

output result contains a merged foreign key sub-graph so that the misinterpretation problem can be avoided. However, the repetitive constraint data generation problem still exists (Figure 4.10).

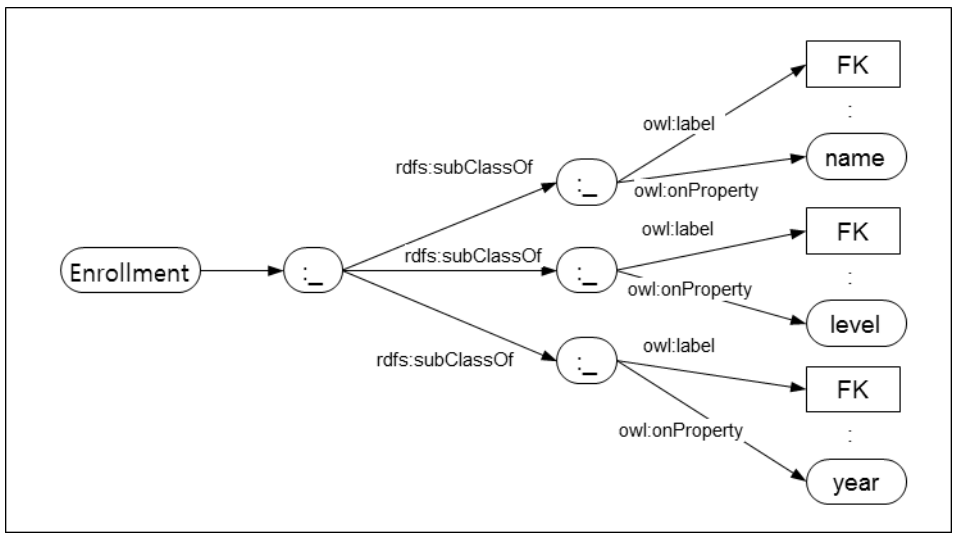



**Figure 4.10.** An example of transformation using grouped foreign key rule.

Thus, optimized the foreign key rule is defined as follows:

**Optimized foreign key rule:** FKeyAttr(*fkey*, *r*, *s*) ∧ subClassOf(*r*, *_b*) ∧ MinCard(*fkey*, *_b*, *1*) ∧ type(*A*, *fkey*) → FK(*A*, *r*, *s*),

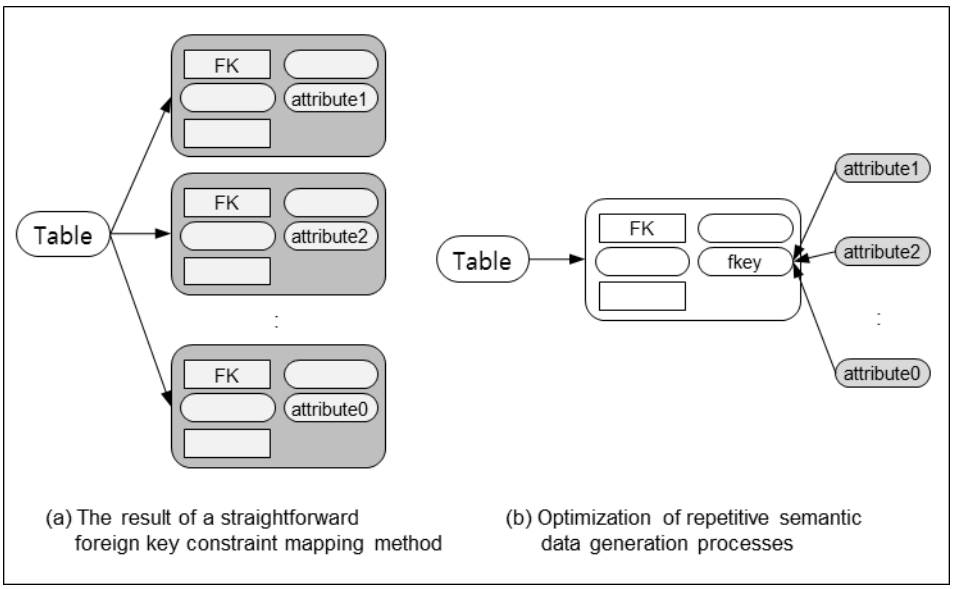



**Figure 4.11.** Transformation of a multi-column foreign key using optimized foreign key rule.

where *A* is a set of attributes which a foreign key is composed of, *r* is a relational table that contains *A*, *s* is a table referenced by table *r*, *fkey* is an OWL property to represent a foreign key constraint of *r*, *_b* is a blank node. The predicates on the left hand side are defined in section 4.2, and the rule specifies foreign key defined by MinCard(*fkey*, *_b*, *1*) and type(*A*, *fkey*) to assign attribute set *A* with the foreign key. As shown in Figure 4.11, optimized foreign key rule outputs lessor constraint data. Instead of producing every foreign key constraint sub-graphs (Figure 4.11(a)), optimized foreign key rule only generates a single foreign key constraint, which is linked by foreign key column attributes (Figure 4.11(b)). The rule generates compact output data without repetitive constraints data,

and the output data contains a merged foreign key sub-graph so that the misinterpretation problem can be avoided.

### 4.3.3 Rules for Multi-column Unique

The following rule is for RDB to RDF mapping of unique constraint defined by a single attribute:

**Base unique rule:** NonFKeyAttr($a$, $r$) $\wedge$ IFP($a$) $\wedge$ subClassOf($r$, $\_b$) $\wedge$ MaxCard($a$, $\_b$, $1$) $\wedge$ ($\exists !v$) a($r$, $v$) $\rightarrow$ Unique($a$, $r$),

where $a$ is an attribute of relational table $r$, $\_b$ is a blank node, $v$ is an attribute value. The predicates on the left hand side are defined in section 4.2, and the rule defines a predicate with a unique existential quantifier ($\exists !v$) a($r$, $v$) such that there is only one attribute value $v$ contained in the domain of a($r$, $v$). If the base unique rule is used to transform a single column unique constraint, then a single sub-graph expressing an unique constraint is generated by the mapping rule (Figure 4.12). On the other hand, if the base unique rule is used to transform a table with a unique constraint that is comprised of three attributes, then three sub-graphs expressing the unique constraint is generated by the mapping rule (Figure 4.13). The output result not only contains repetitive sub-graphs of constraint data, but can also be misinterpreted as the table has three individual unique constraints.

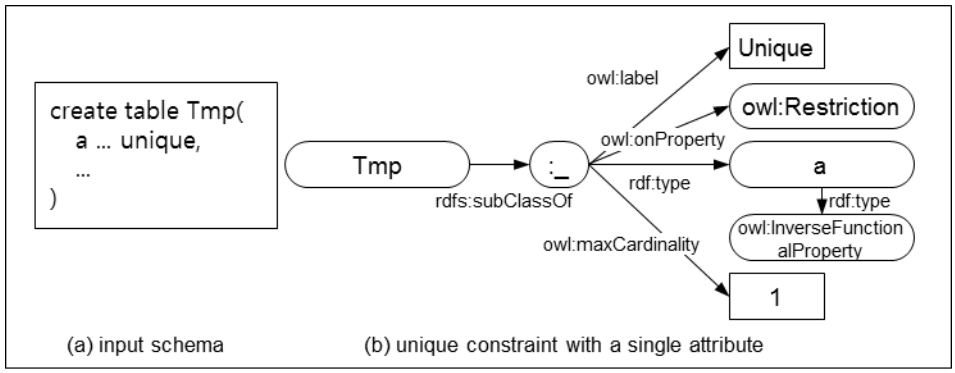



**Figure 4.12.** An example of mapping a single column unique constraint.

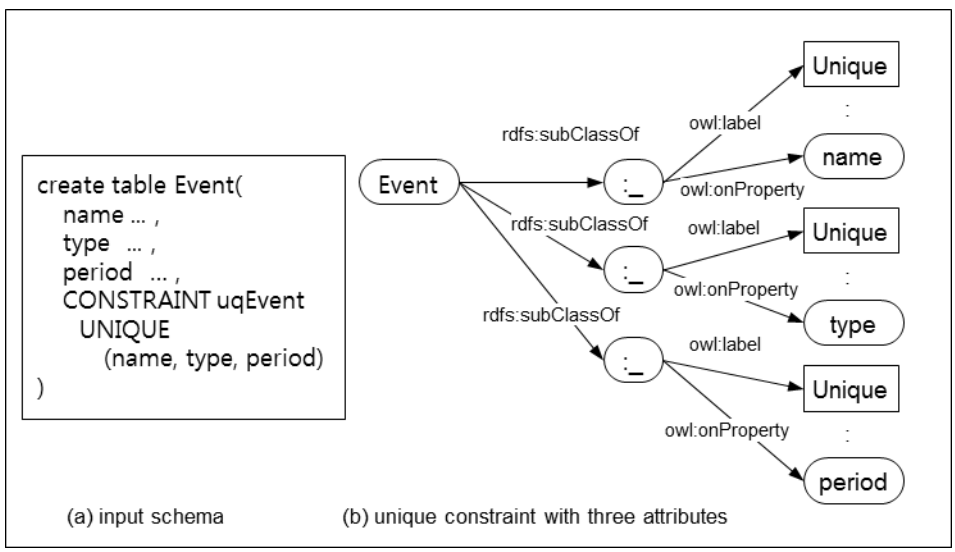



**Figure 4.13.** Transformation of a multi-column unique using base unique rule.

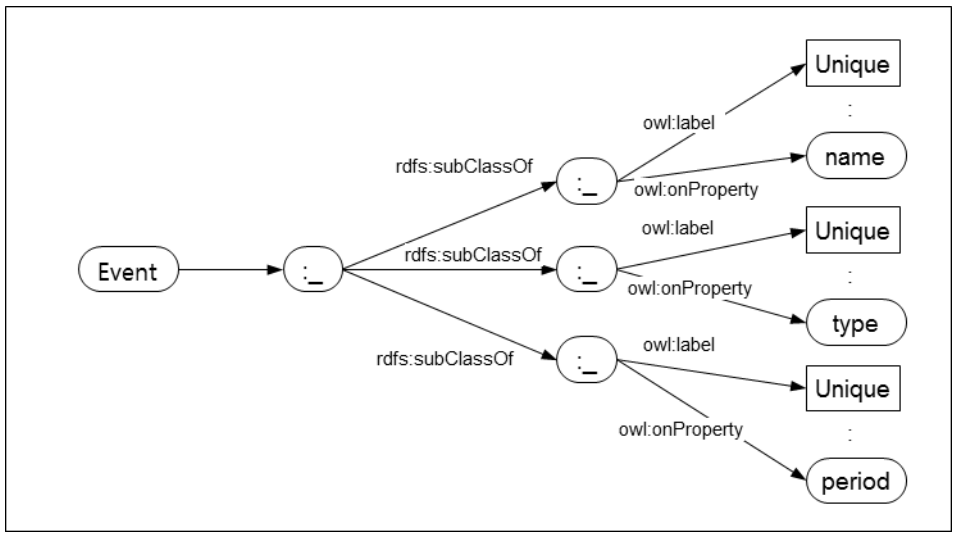



**Figure 4.14.** An example of transformation using grouped unique rule.

To avoid the misinterpretation problem, grouped unique rule is defined as follows:

**Grouped unique rule:** NonFKeyAttr(*A*, *r*) ∧ IFP(*A*) ∧ subClassOf(*r*, *_g*) ∧ subClassOf(_g, *_B*) ∧ MaxCard(*A*, *_B*, *1*) ∧ (∃!*V*) A(*r*, *V*) → PK(A, *r*),

where *A* is a set of attributes which a unique constraint is composed of, *r* is a relational table that contains *A*, *_B* is a blank node set of unique constraints of *A*, *_g* is a blank node for grouping, *V* is a value set of *A*. The predicates on the left hand side are defined in section 4.2, and the rule specifies unique constraint defined by

MaxCard(*A, _B, 1*) and ($\exists !V$) A(*r, V*) to assign attribute set *A* with a unique constraint. The output result contains a merged unique constraint sub-graph so that the misinterpretation problem can be avoided. However, the repetitive constraint data generation problem still exists (Figure 4.14).

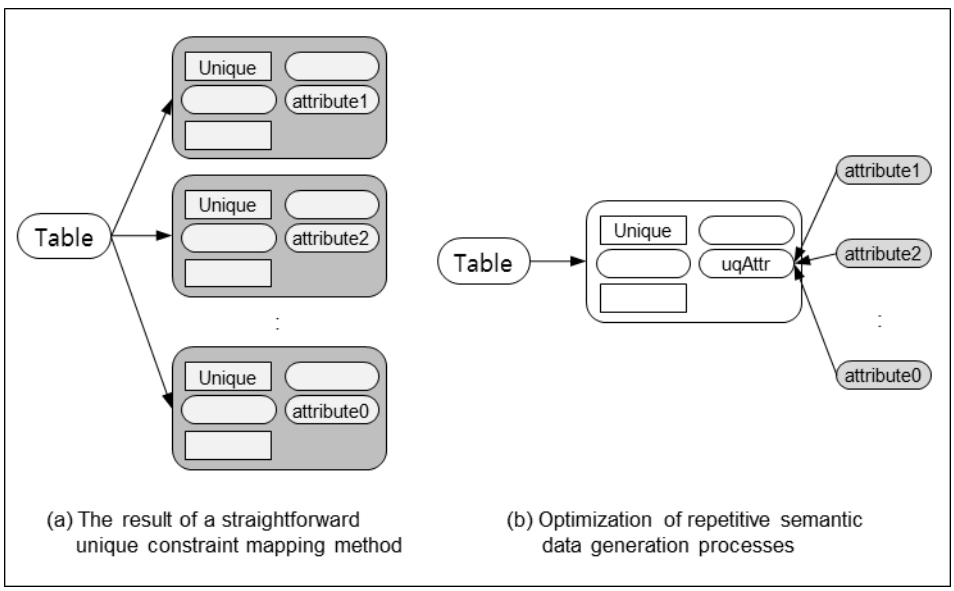



**Figure 4.15**. Transformation of a multi-column unique using optimized unique rule.

Thus, optimized the unique rule is defined as follows:

**Optimized unique rule:** NonFKeyAttr(*uqAttr, r*) ∧ IFP(*uqAttr*) ∧ subClassOf(*r, _b*) ∧ MaxCard(*uqAttr, _b, 1*) ∧ ($\exists !V$)A(*r, V*) ∧ type(*A, uqAttr*) → PK(*A, r*),

where *A* is a set of attributes which a unique constraint is composed of, *r* is a relational table that contains *A*, *uqAttr* is an OWL property to represent unique constraint of *r*, *_b* is a blank node, *V* is a value set of *A*. The predicates on the left hand side are defined in section 4.2, and the rule specifies unique constraint defined by MaxCard(*uqAttr*, *_b*, *1*), ($\exists !V$)A(*r*, *V*) and type(*A*, *uqAttr*) to assign attribute set *A* with the unique constraint. As shown in Figure 4.15, optimized unique rule outputs lessor constraint data. Instead of producing every unique constraint sub-graphs (Figure 4.15(a)), optimized unique rule only generates a single unique constraint, which is linked by unique column attributes (Figure 4.15(b)). The output result contains a merged unique constraint sub-graph so that the misinterpretation problem can be avoided. Moreover, the rule generates compact output data without repetitive constraint data.

# 4.4 Mapping Multiple Keys

In the previous section, rules for mapping multi-column keys are provided. On the other hand, as a single relational table is able to contain two or more foreign keys or unique constraints, optimized rules for mapping multiple keys are defined to reduce output data size in this section.

## 4.4.1 Mapping Multiple Foreign Keys

Suppose a table contains three foreign keys, the first foreign key is composed of

three attributes, the second foreign key is composed of two attributes, and the third foreign key is composed of a single attribute (Figure 4.16).

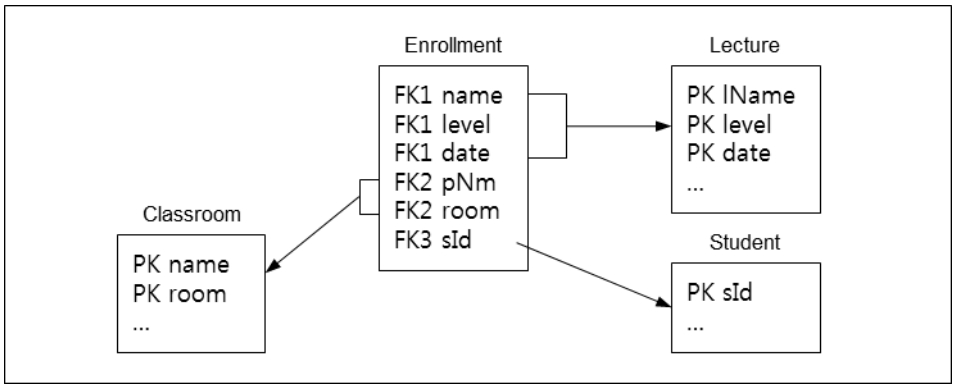



**Figure 4.16.** An example schema for mapping multiple foreign keys.

The number of generated foreign key constraint semantic sub-graphs is six (the number of used attributes) by naïve mapping rule. On the other hand, the number of generated foreign key constraint semantic sub-graphs is three (the number of foreign keys) by optimized foreign key rule described in section 4.2.2. Given a list of foreign key attribute sets $A_i = \{A_1, A_2, \ldots, A_n\}$, mapping rule of multiple foreign keys based on the rule described in section 4.2.2, as follows:

**Base multiple foreign key rule:** $\text{FKeyAttr}(fkey_i, r, s_i) \wedge \text{subClassOf}(r, \_b_i) \wedge \text{MinCard}(fkey_i, \_b_i, 1) \wedge \text{type}(A_i, fkey_i) \rightarrow \text{FK}(A_i, r, s_i)$,

where $A_i$ is a set of attributes which i-th foreign key is composed of, $r$ is a relational table that contains $A_i$, $s_i$ is a table referenced by table $r$, $fkey_i$ is an OWL property to represent a foreign key constraint of $r$, $\_b_i$ is a blank node. The predicates on the left hand side are defined in section 4.2, and the rule specifies foreign key defined by MinCard($fkey_i$, $\_b$, *1*) and type($A_i$, $fkey_i$) to assign attribute set $A_i$ with each foreign key.

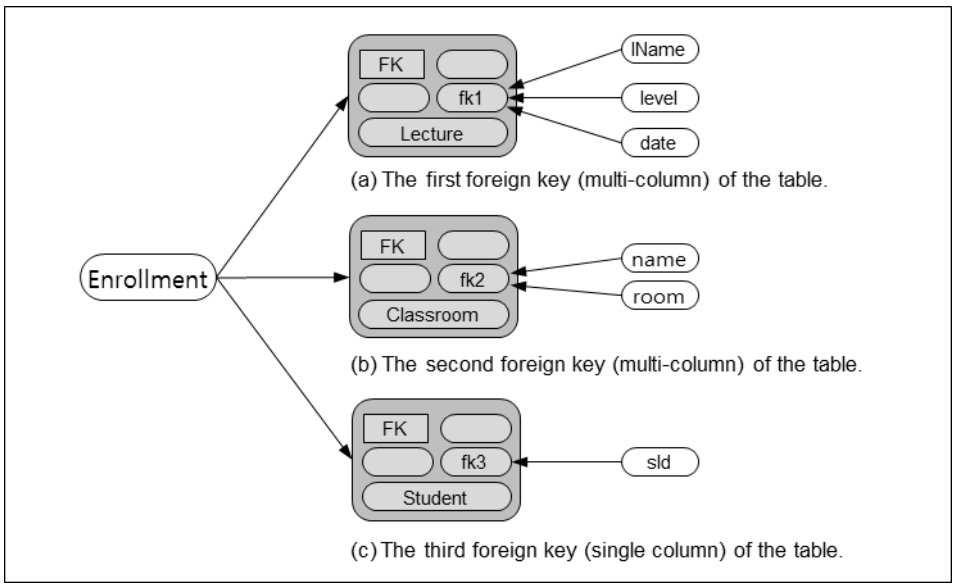



**Figure 4.17.** An example of transformation using base multiple foreign key rule.

Figure 4.17 shows a mapping result using base multiple foreign key rule. The number of foreign key constraint sub-graphs is reduced than the one by naïve mapping rule. However, repetitive foreign key constraint sub-graphs still exist in

the result of the base multiple foreign key rule, which are to be optimized to reduce repetitive data generation.

Given a list of foreign key attribute sets $A_i = \{A_1, A_2, \ldots, A_n\}$, optimized mapping rule of multiple foreign keys, as follows:

**Optimized multiple foreign key rule:** FKeyAttr(*fkConstraint*, *r*, $s_i$) ∧ subClassOf(*r*, *_b*) ∧ MinCard(*fkConstraint*, *_b*, *1*) ∧ type($fkey_i$, *fkConstraint*) ∧ Range($fkey_i$ , $s_i$) ∧ type($A_i$, $fkey_i$) → FK($A_i$, *r*, $s_i$),

where $A_i$ is a set of attributes which i-th foreign key is composed of, *r* is a relational table that contains $A_i$, $s_i$ is a table referenced by table *r*, *fkConstraint* is an OWL property to represent a foreign key constraint of *r*, $fkey_i$ is an OWL property to be typed of *fkConstraint* and to be defined as each foreign key of *r*, $\_b_i$ is a blank node. The predicates on the left hand side are defined in section 4.2, and the rule specifies foreign key defined by MinCard(*fkConstraint*, *_b*, *1*) and type($fkey_i$, *fkConstraint*) to assign $fkey_i$ with the foreign key constraint. Range($fkey_i$ , $s_i$) is a predicate to assign $s_i$ as domain value of property $fkey_i$. All constraint information of $fkey_i$ is inherited to $A_i$ by predicate type($A_i$, $fkey_i$).

Figure 4.18 shows a mapping result using optimized multiple foreign key rule. The number of foreign key constraint sub-graphs is obviously reduced than the one by base multiple foreign key rule. The rule generates only a single foreign key

constraint sub-graph (Figure 4.18(a)) and multiple foreign key semantics is expressed as a simplified sub-graph structure (Figure 4.18(b)).

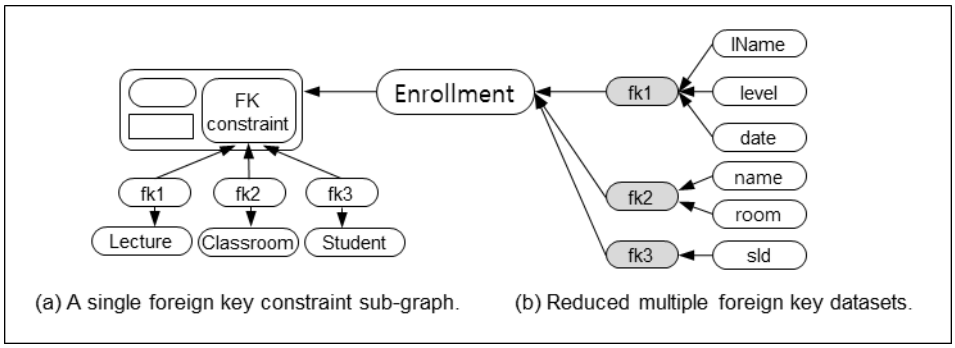



**Figure 4.18.** An example of transformation using optimized multiple foreign key rule.

## 4.4.2 Mapping Multiple Unique

Suppose a table contains three unique constraints, the first one is composed of three attributes, the second one is composed of two attributes, and the third one is composed of a single attribute (Figure 4.19).

The number of generated unique constraint semantic sub-graphs is six (the number of used attributes) by naïve mapping rule. On the other hand, the number of generated unique constraint semantic sub-graphs is three (the number of unique constraints) by optimized unique rule described in section 4.2.3. Given a list of unique attribute sets $A_i = \{A_1, A_2, \ldots, A_n\}$, mapping rule of multiple unique constraints based on the rule described in section 4.2.3, as follows:

```
create table Member(
   ... ,
   CONSTRAINT uq1 UNIQUE (name, type, code)
   CONSTRAINT uq2 UNIQUE (SSN)
   CONSTRAINT uq3 UNIQUE (alias, code)
   ...
)
```

**Figure 4.19.** An example schema for mapping multiple unique constraints.

**Base multiple unique rule:** $\text{NonFKeyAttr}(uq_i, r) \wedge \text{IFP}(uq_i) \wedge \text{subClassOf}(r, \_b_i) \wedge \text{MaxCard}(uq_i, \_b_i, 1) \wedge (\exists !V_i)\ \text{A}_i(r, V_i) \wedge \text{type}(A_i, uq_i) \rightarrow \text{Unique}(A_i, r)$,

where $A_i$ is a set of attributes which i-th unique constraint is composed of, $r$ is a relational table that contains $A_i$, $uq_i$ is an OWL property to represent a unique constraint of $r$, $\_b_i$ is a blank node, $V_i$ is a value set of $A_i$. The predicates on the left hand side are defined in section 4.2, and the rule specifies unique constraint defined by $\text{MaxCard}(uq_i, \_b, 1)$, $(\exists !V_i)\ \text{A}_i(r, V_i)$, and $\text{type}(A_i, uq_i)$ to assign attribute set $A_i$ with each unique constraint.

Figure 4.20 shows a mapping result using the base multiple unique rule. The number of unique constraint sub-graphs is reduced than the one by naïve mapping rule. However, repetitive unique constraint sub-graphs still be produced by the base

multiple unique rule. Thus, the base multiple unique rule should be optimized to avoid the repetitive data generation.

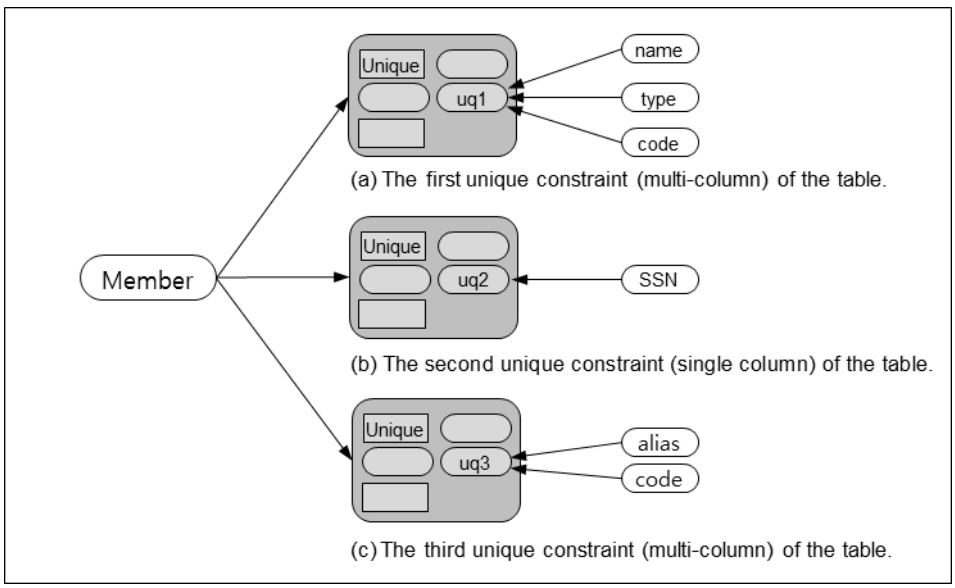



**Figure 4.20.** An example of transformation using base multiple unique rule.

Given a list of unique attribute sets $A_i = \{A_1, A_2, \ldots, A_n\}$, optimized mapping rule of multiple unique constraints, as follows:

**Optimized multiple unique rule:** NonFKeyAttr(*uqConstraint*, *r*) $\wedge$ IFP(*uqConstraint*) $\wedge$ subClassOf(*r*, *_b*) $\wedge$ MaxCard(*uqConstraint*, *_b*, *1*) $\wedge$ $(\exists !V_i)$ $A_i(r, V_i)$ $\wedge$ type($uq_i$, *uqConstraint*) $\wedge$ type($A_i$, $uq_i$) $\rightarrow$ Unique($A_i$, *r*),

where $A_i$ is a set of attributes which i-th foreign key is composed of, $r$ is a relational table that contains $A_i$, *uqConstraint* is an OWL property to represent a unique constraint of $r$, $uq_i$ is an OWL property to be typed of *uqConstraint* and to be defined as each unique constraint of $r$, *_b* is a blank node. The predicates on the left hand side are defined in section 4.2, and the rule specifies unique constraint defined by MaxCard(*uqConstraint, _b, 1*) and ($\exists !V_i$) $A_i(r, V_i)$. The predicate type($uq_i$, *uqConstraint*) is to assign $uq_i$ with the unique constraint. All constraint information of $uq_i$ is inherited to $A_i$ by predicate type($A_i$, $uq_i$).

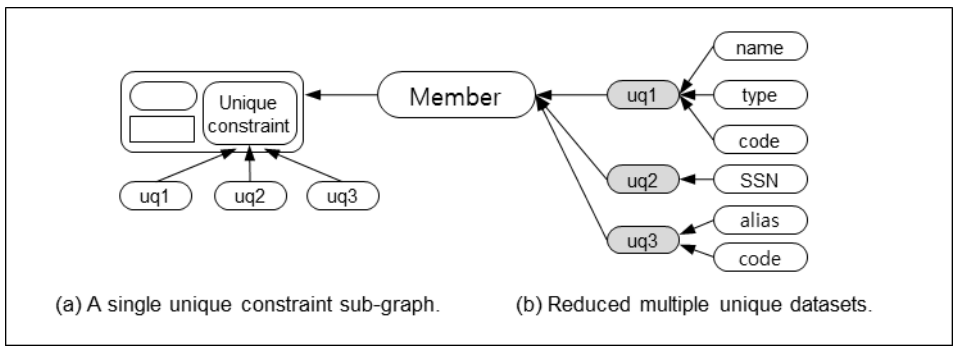



**Figure 4.21.** An example of transformation using optimized multiple unique rule.

Figure 4.21 shows a mapping result using optimized multiple unique rule. The number of unique constraint sub-graphs is obviously reduced than the one by base multiple unique rule. The rule generates only a single unique constraint sub-graph (Figure 4.21(a)) and multiple unique semantics is expressed as a simplified sub-graph structure (Figure 4.21(b)).

# 4.5 Relational Meta-schema Vocabulary

In this section, a semantic vocabulary set called relational meta-schema vocabulary (RMSV) is provided. RMSV contains relational resources that are used to transform relational input data. As RMSV initially defines relational concepts and relationships among the relational concepts, mapping rules that use RMSV can generate compact and intuitive semantic relational output data. Optimized mapping rules based on RMSV are also provided.

RMSV 1 defines a type of a relational table and relational tables containing integrity constraints (Figure 4.22). The type of a relational table “Relation” in RMSV 1 is used in Rule 1, which transforms input table data into semantic table resources, as follows:

**Rule 1:** Rel($r$) $\wedge$ $\neg$BinRel($r, a1 \ldots m, s, b1 \ldots n, t$) $\rightarrow$ Relation($r$),

where the predicates used on the left hand side are defined in section 4.2, and Relation($r$) is a predicate that verifies if $r$ is a relational table and not a binary relation.

The relational tables containing integrity constraints are defined as NotNullConstrainedRelation, UniqueConstrainedRelation, PKConstrainedRelation, FKConstrainedRelation, DefaultConstrainedRelation, and CheckConstrained-

Relation in RMSV 1. The constrained relations are used in Rule 6 to Rule 11 to transform input integrity constraints into semantic integrity constraint resources.

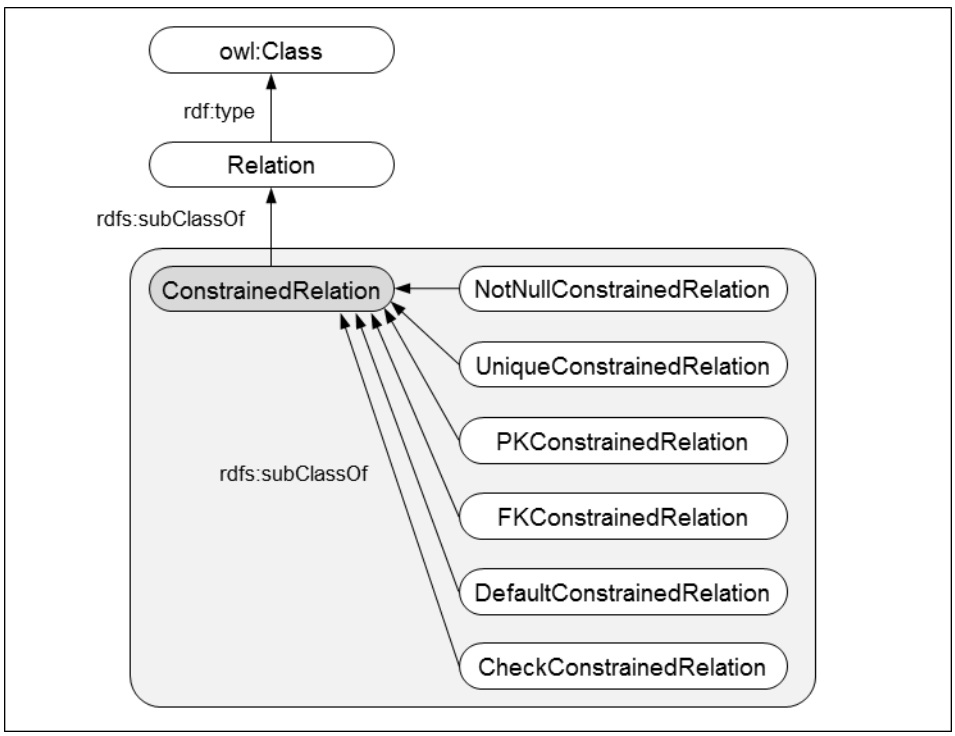



**Figure 4.22.** [RMSV 1] Relation and constrained relation.

RMSV 2 defines a type of a relational attributes and relational attributes containing integrity constraints (Figure 4.23). The type of a relational attribute "Attribute" in RMSV 2 is used in Rule 2, which transforms input attribute data into semantic attribute resources, as follows:

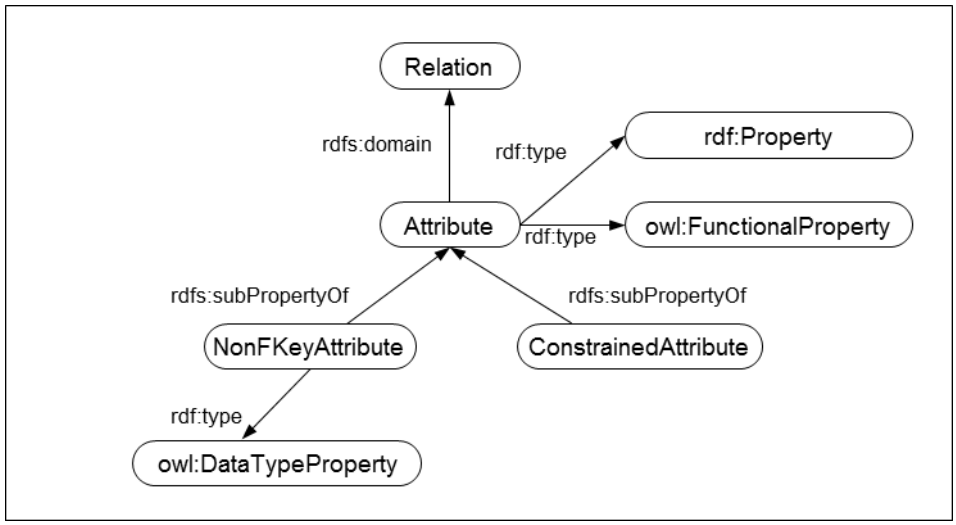



**Figure 4.23.** [RMSV 2] Attribute, non-foreign key attribute, and Constrained Attribute.

**Rule 2:** Prop($a$, $r$, _) $\wedge$ FP($a$) $\rightarrow$ Attribute($a$, $r$),

where the predicates on the left hand side are defined in section 4.2, and the predicates on the right hand side represent transforms of a relational attribute $a$.

NonFKeyAttribute is used to specify that target attribute is not a foreign key, and ConstrainedAttribute is used in Rule 6 to Rule 11 to transform input integrity constraints into semantic integrity constraint resources.

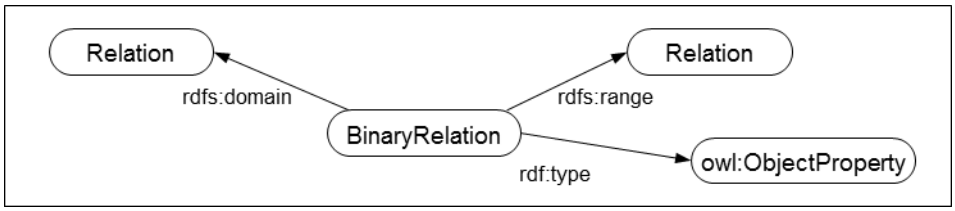



**Figure 4.24.** [RMSV 3] Binary relation.

RMSV 3 defines a type of a binary relation (Figure 4.24). The resource BinaryRelation in RMSV 3 is used in Rule 3, which transforms input binary relation data into semantic binary relation resources, as follows:

**Rule 3:** BinRel(*r, a1…m, s, b1…n, t*) ∧ ¬BinRel(*s*, _, _, _, _) ∧ ¬BinRel(*t*, _, _, _, _) → BinaryRelation(*r, s, t*),

where the predicates on the left hand side are defined in section 4.2, and BinaryRelation(*r, s, t*) is a predicate that verifies if a binary relation *r* can be transformed into semantic resource *BinaryRelation* (typed OWL object property), which is a semantic vocabulary to notate binary relations.

RMSV 4 and RMSV 5 define types of an identifying relationship and a non-identifying relationship (Figure 4.25 and 4.26). The resources Identifying relationship and non-identifying relationship in RMSV 4 and RMSV 5 are used in Rule 4 and Rule 5, as follows:

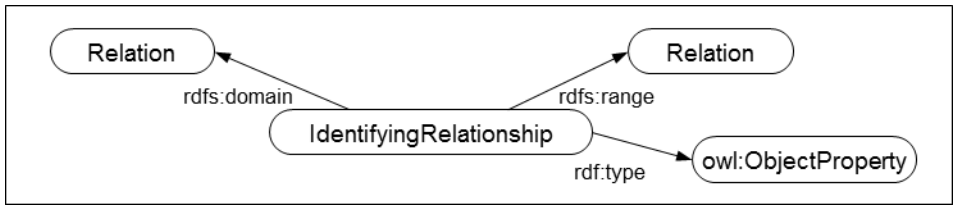



**Figure 4.25.** [RMSV 4] Identifying relationship.

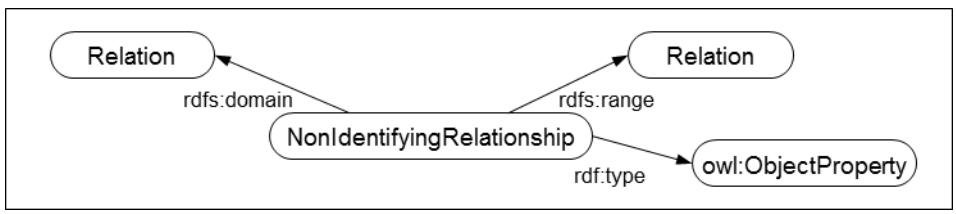



**Figure 4.26.** [RMSV 5] Non-identifying relationship.

**Rule 4:** Rel($s$) ∧ Rel($t$) ∧ PK($a$, $s$) ∧ FK($a$, $s$, _, $t$) ∧ ObjProp($r$, $s$, $t$) ∧ FP($r$) → IdentifyingRelationship($r$, $s$, $t$)

**Rule 5:** Rel($s$) ∧ Rel($t$) ∧ PK($a$, $s$) ∧ ¬FK($a$, $s$,_,$t$) ∧ ObjProp($r$, $s$, $t$) ∧ FP($r$) → NonIdentifyingRelationship($r$, $s$, $t$),

where the predicates on the left hand side are defined in section 4.2, IdentifyingRelationship($r$, $s$, $t$) is a predicate that verifies identifying relationships,

and NonIdentifyingRelationship(*r, s, t*) is a predicate that verifies non-identifying relationships.

RMSV 6, 7, and 8 define semantic resources to transform integrity constraints and used in Rule 6 to Rule 11 (rules of mapping not null, unique, primary key, foreign key, default, and check constraint), as follows:

**Rule 6**: hasConstraint($r$, $\_nn$) $\wedge$ ComposedOf($\_nn$, $a$) $\wedge$
NotNullConstrainedRelation($\_nn$) $\wedge$ NotNullConstrainedAttribute($a$) $\rightarrow$
NotNull($a$, $r$)

**Rule 7**: hasConstraint($r$, $\_uq_i$) $\wedge$ ComposedOf($\_uq_i$, $A_i$) $\wedge$
UniqueConstrainedRelation($\_uq_i$) $\wedge$ UniqueConstrainedAttribute($A_i$) $\rightarrow$
Unique($A_i$, $r$)

**Rule 8**: hasConstraint($r$, $\_pk$) $\wedge$ ComposedOf($\_pk$, $A$) $\wedge$
PKConstrainedRelation($\_pk$) $\wedge$ PKConstrainedAttribute($A$) $\rightarrow$ PK($A$, $r$)

**Rule 9**: hasConstraint($r$, $\_fk_i$) $\wedge$ ComposedOf($\_fk_i$, $A_i$) $\wedge$
FKConstrainedRelation($\_fk_i$) $\wedge$ FKConstrainedAttribute($A_i$) $\rightarrow$ FK($A_i$, $r$)

**Rule 10**: hasConstraint(*r*,*_df*) ∧ ComposedOf(*_df*, *a*) ∧ DefaultValue(*a*,*v*) ∧ DefaultConstrainedRelation(*_df*) ∧ DefaultConstrainedAttribute(*a*) → Default(*a*, *r*)

**Rule 11**: hasConstraint(*r*,*_ck*) ∧ ComposedOf(*_ck*,*a*) ∧ CheckCondition(*a*,*v*) ∧ CheckConstrainedRelation(*_ck*) ∧ CheckConstrainedAttribute(*a*) → Check(*a*, *r*)

where *r* is a relational table. The blank nodes *_nn*, *_uq*, *_pk*, *_fk*, and *_df* are used as root nodes of each constraint sub-graphs ($_uq_i$ and $_fk_i$ are for multiple unique constraints and multiple foreign keys). *a* is an attribute, *A* is a set of attributes for multi-column constraints (primary key, foreign key, unique constraint), and $A_i$ is a series of attribute sets for multiple constraints (multiple foreign keys and multiple unique constraints). *v* in Rule 10 is a default value and *v* in Rule 11 is a check condition.

RMSV 6 defines NotNullConstrainedRelation, UniqueConstrainedRelation, PKConstrainedRelation, FKConstrainedRelation, DefaultConstrainedRelation, and CheckConstrainedRelation that are relations containing each integrity constraint (Figure 4.27, 4.28, and 4.29). For instance, if attribute *a* in table *r* is defined by not null constraint, then both *a* and *r* are linked to blank node *_nn*. The blank node *_nn* is typed by NotNullConstraintRelation and assigned as a root node of not null constraint semantic sub-graph.

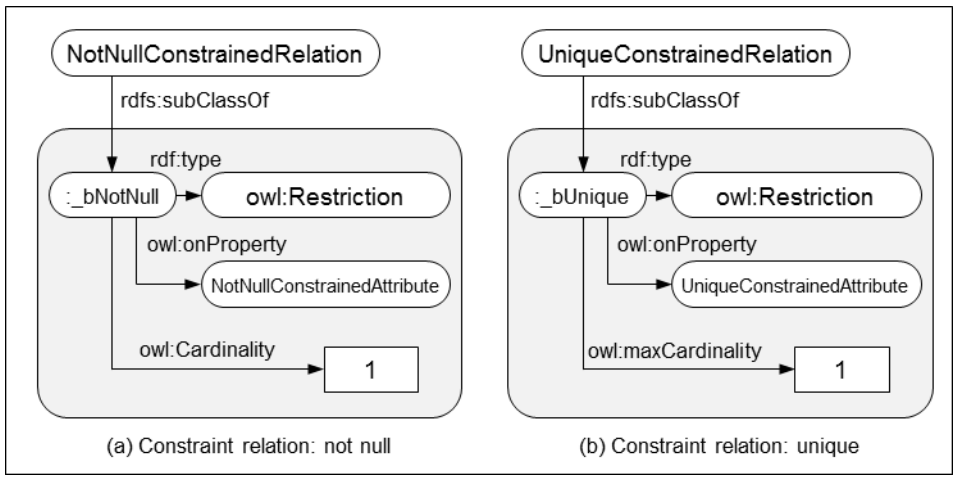



**Figure 4.27.** [RMSV 6-1] Relations that contains integrity constraints.

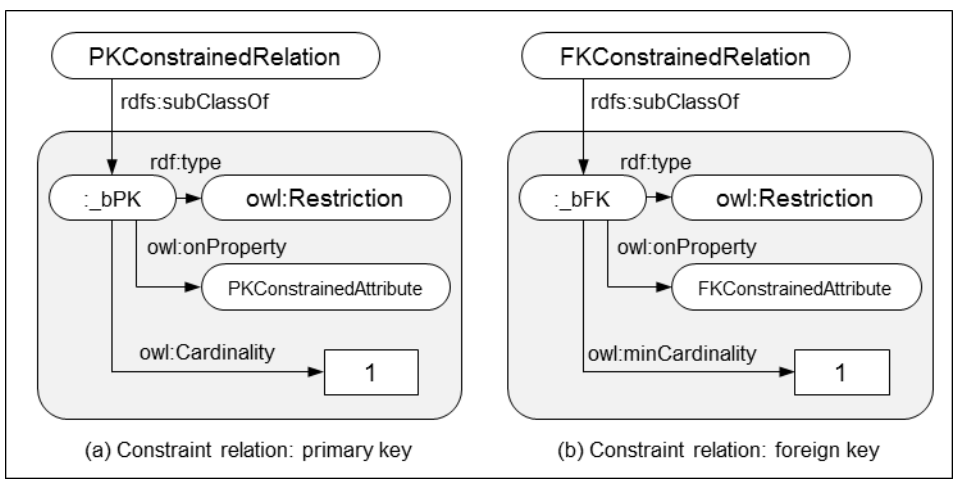



**Figure 4.28.** [RMSV 6-2] Relations that contains integrity constraints.

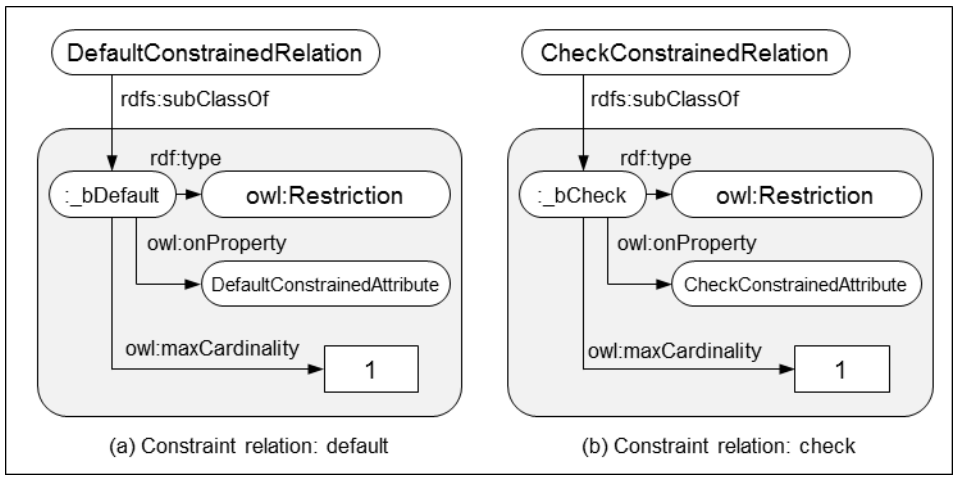



**Figure 4.29.** [RMSV 6-3] Relations that contains integrity constraints.

RMSV 7 defines NotNullConstrainedAttribute, UniqueConstrainedAttribute, PKConstrainedAttribute, FKConstrainedAttribute, DefaultConstrainedAttribute, and CheckConstrainedAttribute that are attributes containing each integrity constraint (Figure 4.30). For example, if attribute *a* in table *r* is defined by default constraint with a default value *v*, then *a* is typed by NotNullConstrainedAttribute and linked to default value *v* using a predicate DefaultValue(*a*, *v*).

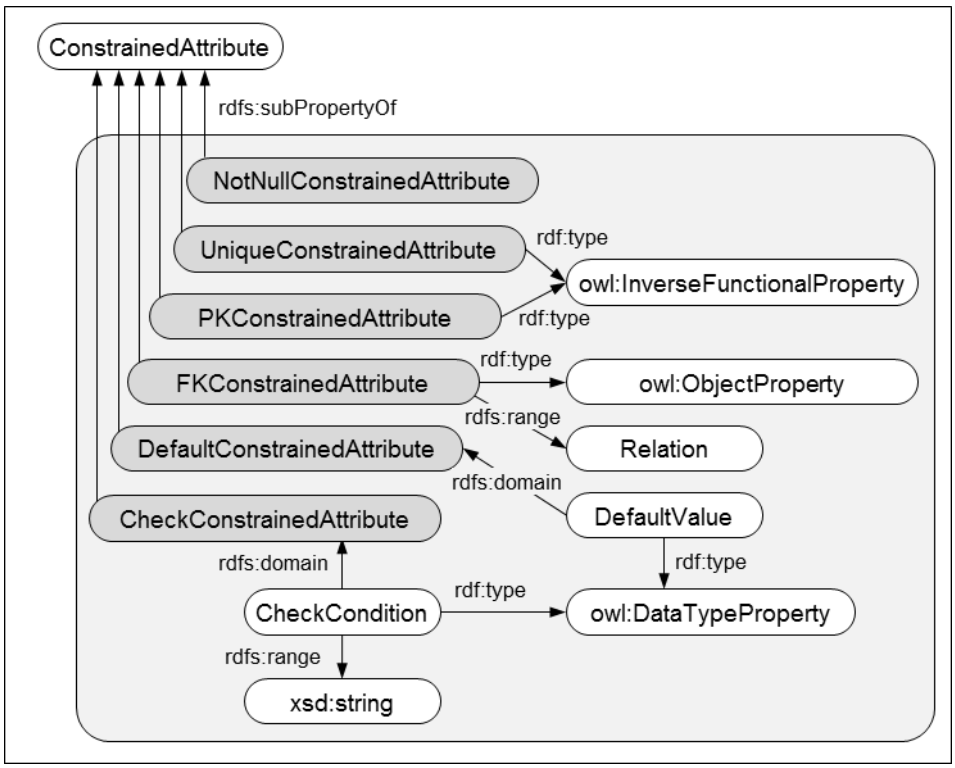



**Figure 4.30.** [RMSV 7] Attributes defined by integrity constraints.

RMSV 8 defines predecates hasConstraint and ComposedOf (Figure 4.31 and 4.32). For instance, if attribute *a* in table *r* is defined by not null constraint, then both *r* is linked to blank node *_nn* using hasConstraint(*r*, *_nn*) and *_nn* is linked to attribute a using ComposedOf(*_nn*, *a*).

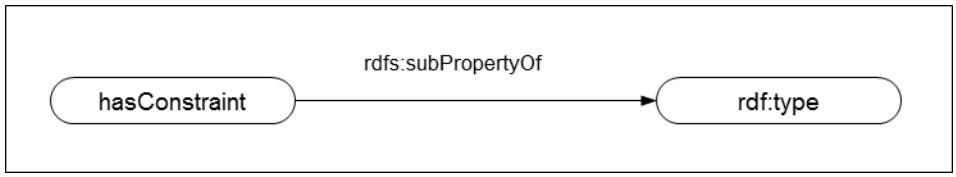



**Figure 4.31.** [RMSV 8-1] The predicate 'hasConstraint' used by relations to assign constraints.

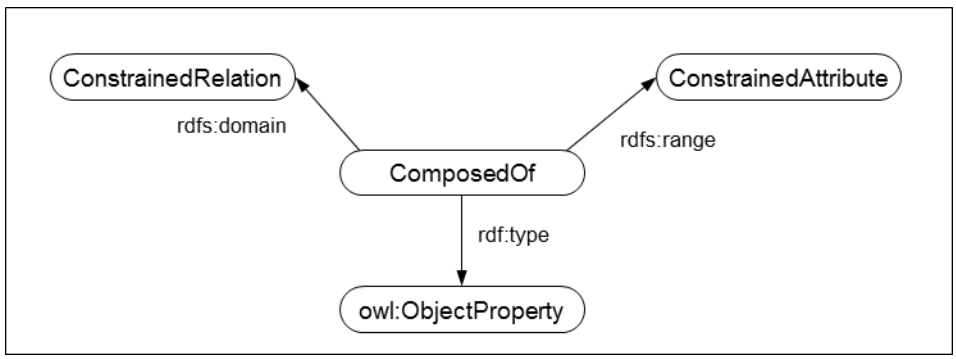



**Figure 4.32.** [RMSV 8-2] The predicate 'ComposedOf' used by between constrained relations and constrained attributes.

Figure 4.33 illustrates an example schema that contains a set of integrity constraints. The transformed semantic outputs by the optimized mapping rule using RMSV is described in Figure 4.34 and by the base mapping rule is in Figure 4.35. The base mapping rule generates every integrity constraint sub-graphs to map constraints in the input schema. On the other hand, the RMSV based optimized mapping rule generates a more optimized graph structure than the base mapping rule. As RMSV initially defines relational concepts including integrity constraints,

the mapping rule using RMSV can generate compact and intuitive semantic output data.

```
create table Product(
   ...
   size        ...  NOT NULL,
   color       ...  NOT NULL,
   regDate     ...  DEFAULT now( ),
   ...
   CONSTRAINT pkP PRIMARY KEY (type, name, date)
   CONSTRAINT fkM FOREIGN KEY (mId) REFERENCES Manufacturer (mId)
   CONSTRAINT fkS FOREIGN KEY (sCd, sNm) REFERENCES Style (cd, nm)
)
```

**Figure 4.33.** An example schema defined using integrity constraints.

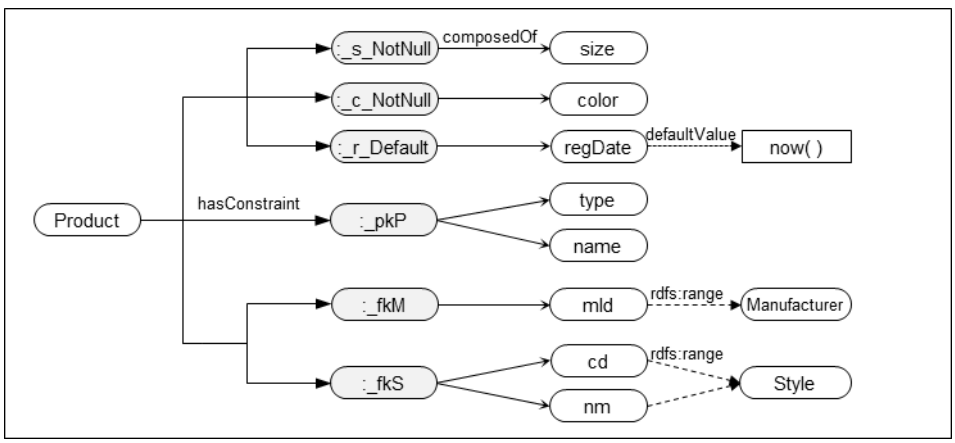



**Figure 4.34.** The RDB2RDF transformation result using the RMSV based optimized mapping rule.

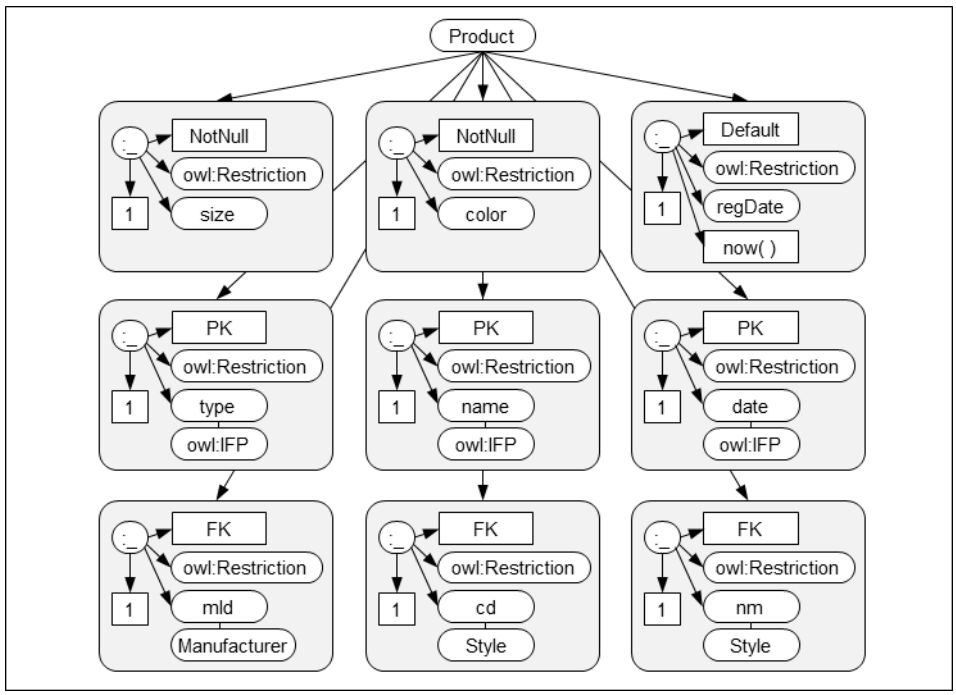



**Figure 4.35.** The RDB2RDF transformation result using the base mapping rule.

# 4.6 Evaluation

Evaluations were conducted on a single node of 3.1 GHz quad-core CPU, 4GB memory, and 2TB hard disk. The experiments are conducted based on five real datasets and a single synthetic dataset. Each real dataset is defined by relational schema information with integrity constraints: Ensembl-compara (DB1), Ensembl

(DB2), PHPmyadmin (DB3), and MusicBrainz (DB4). The DBT2 benchmark was used as a synthetic dataset. A synthetic dataset is generated using DBT2 and the schema of the dataset is restructured by adding integrity constraints to evaluate performance of generating integrity constraints about multi-column keys and multiple keys. The previous approach (OWL ontology based augmented direct mapping) was employed [14] to perform a comparative analysis among direct mapping methods.

In Figure 4.36 and 4.37, we present the key constraints transformation performance of mapping rules. Single column primary keys, multi-column primary keys, single column foreign keys, multi-column foreign keys, multiple foreign keys, single column unique constraints, multi-column unique constraints, and multiple unique constraints are used as input data. The previous approach, base key rule (section 4.2), grouped key rule (section 4.2), optimized key rule (section 4.2), and optimized multiple key rule (section 4.3) are comparatively evaluated.

In Figure 4.36, the horizontal axis represents data size of synthetic key constraint data. The vertical axis is the number of semantic triples as an output data. From the Figure 4.36, our approach generates lesser number of triples compared to the previous method, and optimized multiple key rule generates the smallest output data.

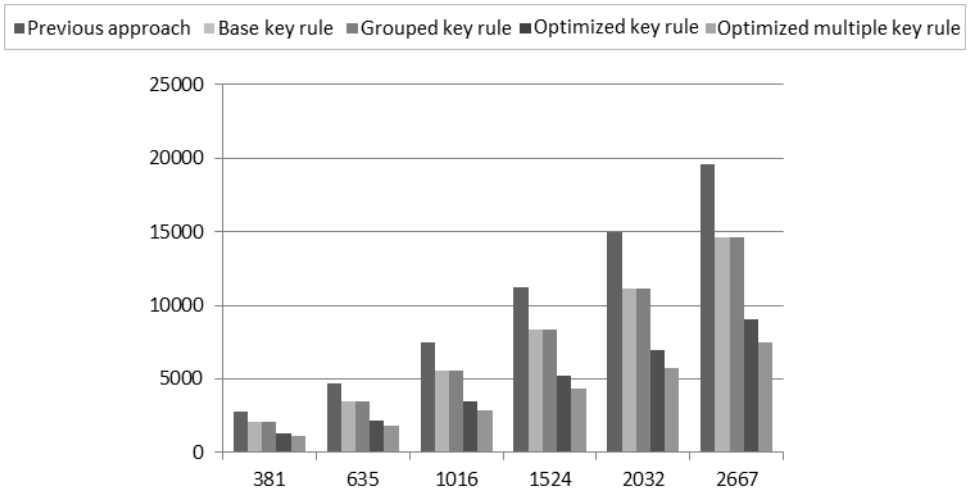



**Figure 4.36.** The number of triples over synthetic multi-column key and multiple key constraints.

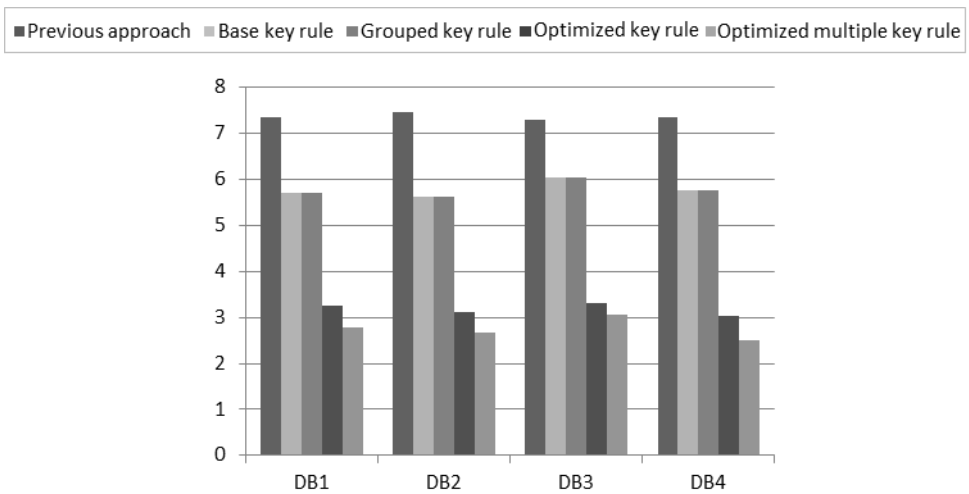



**Figure 4.37.** The average number of triples real multi-column key and multiple key constraints.

Figure 4.37 shows the average number of triples as a result of each transformation method. The horizontal axis represents key constraint data from each real dataset and the vertical axis is the average number of triples generated from transforming a key constraint. The results show that the optimized multiple key rule, which is the most optimized method to reduce repetitive multi-column key and multiple key constraint data, generates compact output data with fewer resources.

In Figure 4.38 and 4.39, we present the performance results of mapping rules. The previous approach, base key rule (section 4.2), optimized key rule (section 4.2), and the optimized rule using RMSV (section 4.4) are comparatively evaluated.

In Figure 4.38, the horizontal axis represents the size of the synthetic relational dataset as input data. The vertical axis is the number of semantic triples as an output data. From the Figure 4.38, the provided approach generates lesser number of triples compared to the previous method, and optimized rule using RMSV generates the smallest output data.

Figure 4.39 shows the average number of triples as a result of each transformation method. The horizontal axis represents each real dataset and the vertical axis is the average number of triples generated from transforming a single relational element. The results show that the optimized rule using RMSV, which initially defines relational concepts including integrity constraints, generates the fewest output data among mapping methods.

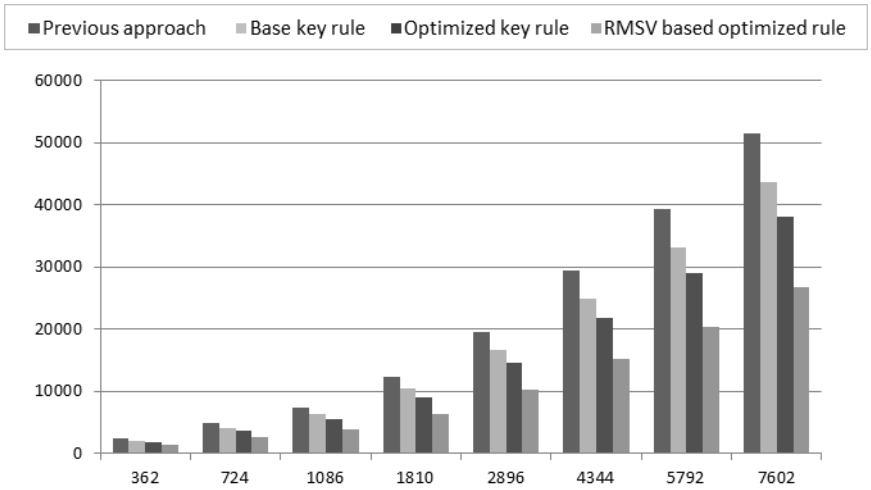



**Figure 4.38.** The number of triples over synthetic relational data.

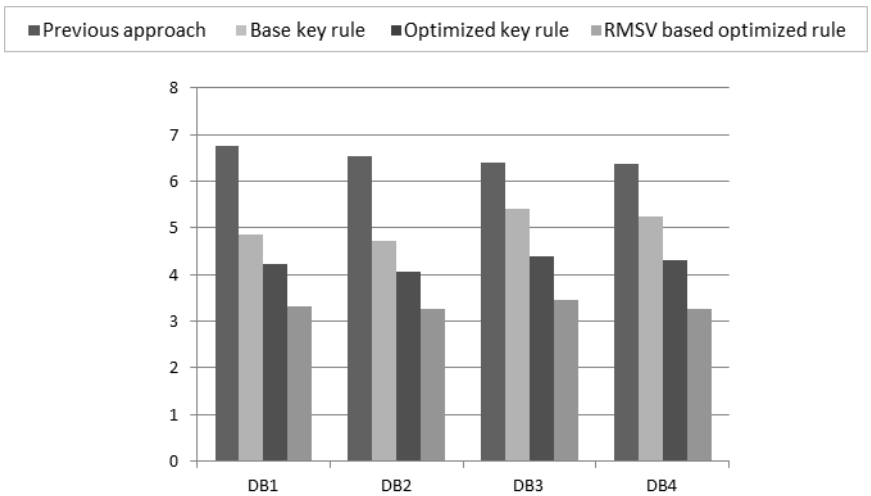



**Figure 4.39.** The average number of triples for a single input real data.

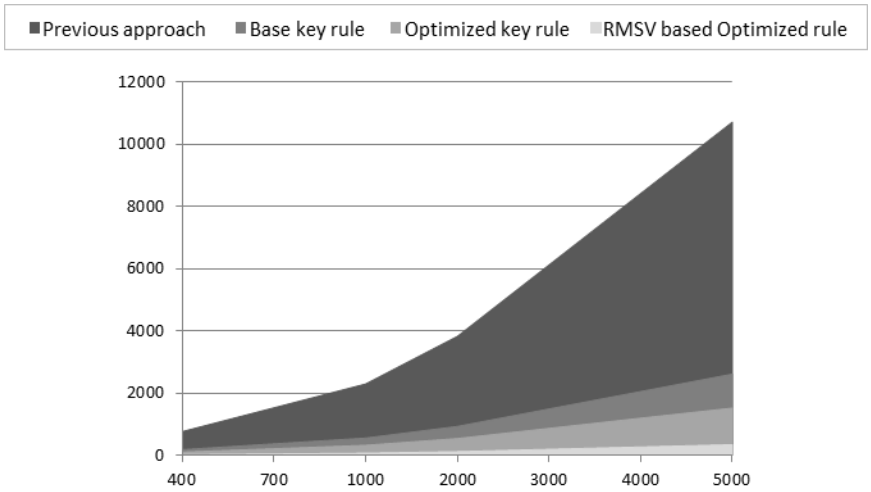



**Figure 4.40.** The number of failed data and duplicated data.

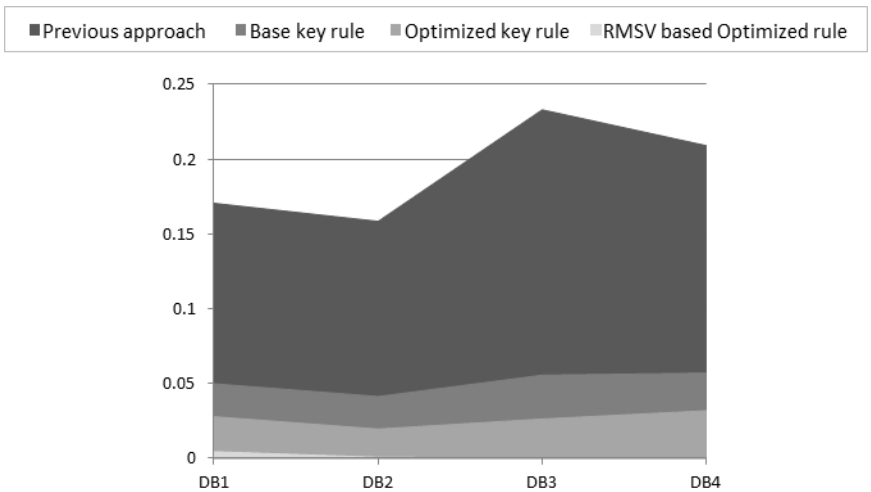



**Figure 4.41.** Transformation failure rate on real datasets.

In Figure 4.40, the horizontal axis represents the size of synthetic relational dataset as input data. The vertical axis is the number of failed semantic data and duplicated output data. The failures of the previous approach result in the incorrect semantic data generation and semantic information loss. Figure 4.41 shows the failure rate of mapping methods in each database. The horizontal axis represents each relational dataset and the vertical axis is the failure rate during the transformation of relational data into semantic data. The previous approach produces the most number of failed outputs as the previous methods lack support in integrity constraints. On the other hand, the provided approach improves the mapping rules and generates lesser false mapping results. Moreover, the optimized rule using RMSV generates multi-column key constraints, multiple keys without information loss and outputs compact and intuitive semantic data using RMSV, which initially defines relational concepts including integrity constraints.

Figure 4.42 and 4.43 show completeness and soundness of mapping rules when input data contains multi-column keys and multiple keys. The horizontal axis represents the size of relational constraints. The vertical axis is recall (Figure 4.42) or precision (Figure 4.43). M1 through M5 are mapping methods for comparative analysis: [90], [64], [67], the method in chapter 4, and the method in this chapter, respectively. The results show that the provided method generates sound and complete mapping results.

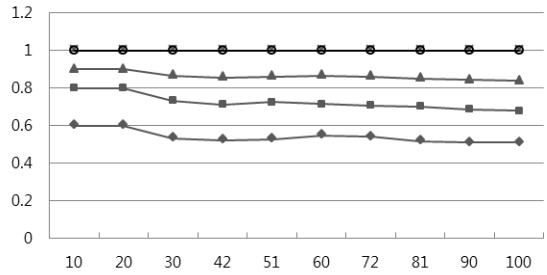



**Figure 4.42.** Completeness of mapping methods on constraints.

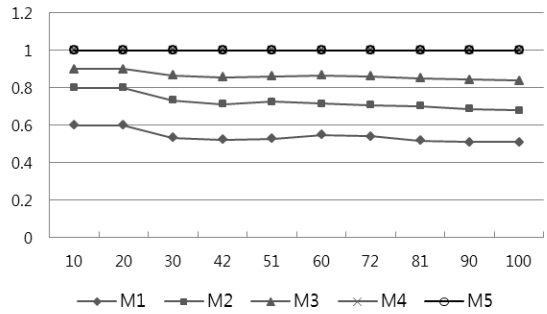



**Figure 4.43.** Soundness of mapping methods on constraints.

**Table 4.2.** Fixed mapping cost of mapping using RMSV.

| Relational concepts | The number of RDF Triples |
|---|---|
| Table | 8 |
| Attribute | 6 |
| Relationships between tables | 9 |
| Constraints | 36 |
| Constraint assertion | 4 |
| Fixed mapping cost | 63 |

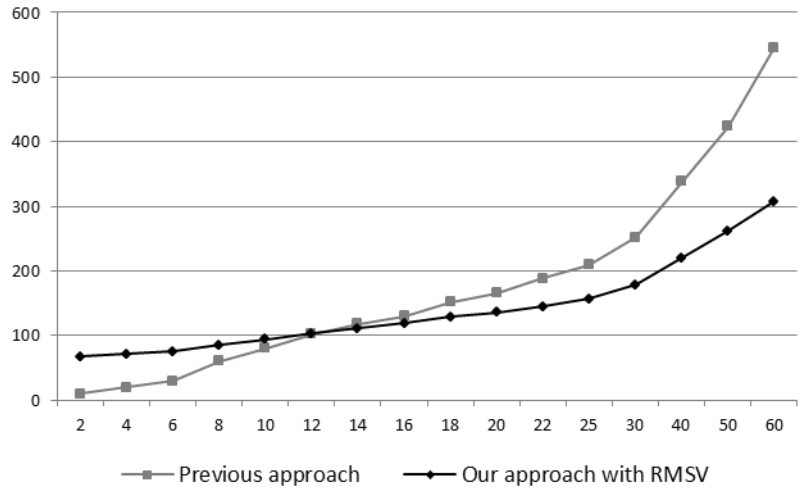



**Figure 4.44.** Mapping cost compared to the previous approach.

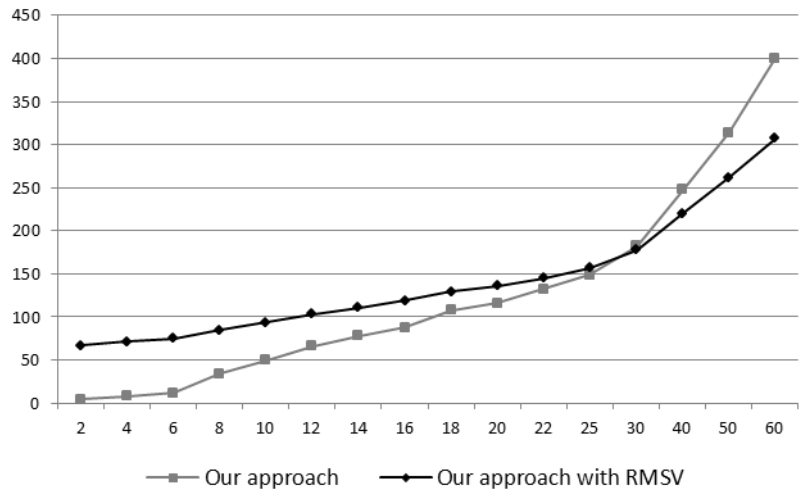



**Figure 4.45.** Mapping cost compared to our approach without RMSV.

Table 4.2 shows a fixed mapping cost of mapping using RMSV. A mapping method with RMSV contains semantic data of relational concepts using 63 RDF triples before transformation. Thus, mapping results initially contain semantic data of relational concepts by RMSV. Figure 4.42 shows that the mapping method with RMSV outputs over 63 RDF triples when input data is small. However, the mapping method with RMSV outputs lesser than previous approach if the size of input data is bigger than 12, which is the same size of two tables where each table contains seven attributes. Figure 4.43 also shows that the mapping method with RMSV performs better than the provided method without RMSV if the size of input data is bigger than 30, which is the size about five tables where each table contains seven attributes and constraints.

# Chapter 5
# Utilization of RDB to RDF Transformation for Information Retrieval

In the previous chapters, RDB to RDF transformation methods that generate semantic data from relational data on the Web are discussed. In this chapter, semantic information utilization in the view of Web information retrieval will be discussed. A semantic data based ranking method is provided. The main idea is improving performance of existing ranking algorithms utilizing semantic data as metadata.

## 5.1 Motivation

In modern Web information retrieval [11-13], PageRank is a representative link-

based ranking method [14, 15]. The authors of PageRank assume that pages with many in-links from other pages are important. In PageRank, each page distributes its rank value to other pages through links among the pages. However, the page rank values are equally distributed without considering the meaning of the links. Due to this feature, unimportant pages containing many in-links could be highly ranked (Figure 5.1).

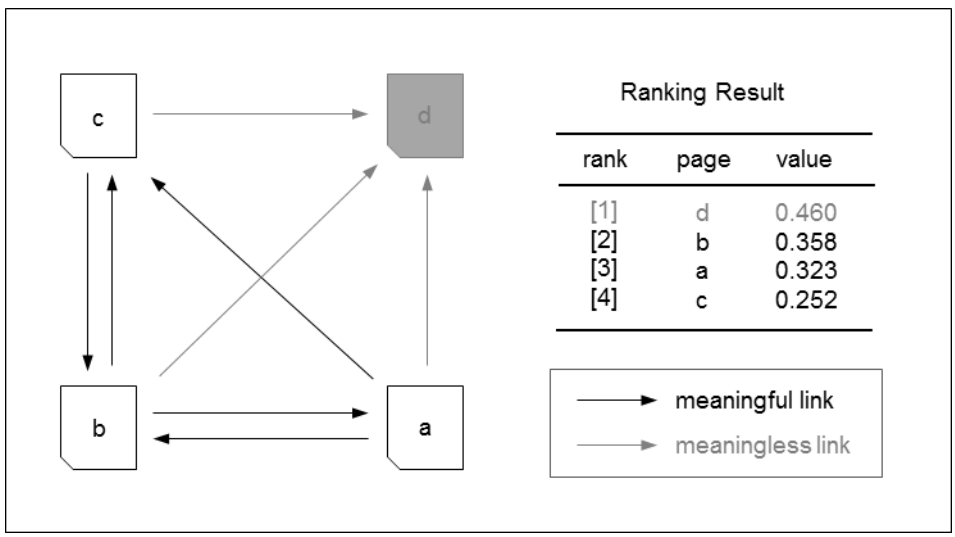



**Figure 5.1.** Limitation of PageRank.

Further studies have tried to improve the rank value propagation [76-80], and some have considered link evaluation. Improved methods using link weighting cause pages containing many meaningless in-links to be low ranked. However, in the current Web structure, links between pages are defined by hyperlink notation. Because hyperlink notation cannot express the meaning of links, it is not easy to

directly evaluate the weight of the links. Therefore, existing works have had to use indirect methods to evaluate the link weights. Moreover, simply analyzing links is insufficient to measure the importance of pages, in that highly ranked pages containing many in-links are not always important, and may even contain meaningless information.

In this chapter, the Weighted Semantic PageRank (WSPR), which evaluates links directly to obtain a more accurate ranking result, is provided. Semantic information is utilized for WSPR, and a semantic Web data structure is formed to manage the semantic information. A semantic Web data structure is created using RDF from existing Web data [81]. Links in a semantic Web data structure contain semantic information and are used to resolve the problem of determining the weight of the links. Thus, the provided ranking method is able to calculate rank values based on the semantic information analyzed from pages and links. Furthermore, WSPR reduces the phenomenon of giving high ranks to unimportant pages. As WSPR uses semantic resources in pages rather than meaningless hyper-links to calculate rank values, it enables important pages to receive high rank values. In addition, the WSPR algorithm is implemented using the Hadoop framework [70] to manage large-scale semantic metadata more effectively.

# 5.2 Previous Work

Search engines answer user queries with ranked page lists created using ranking methods. Early ranking methods were term-based ranking methods that evaluate page importance based on the number of matched terms for a given query [82, 83]. After 1998, alternative link-based ranking methods were provided and demonstrated much higher performance than term-based ranking methods. PageRank [14] and HITS [15] are representative link-based ranking methods. While HITS considers both in-links and out-links to classify pages into authority and hub, PageRank only considers in-links focusing on ranking pages by their popularity. PageRank calculates the rank score as follows:

$$PR(r_i) = d \sum_{j \to i} \frac{1}{N_j} \cdot PR(r_j) + (1 - d), \quad (5.1)$$

where $d$ is a damping factor to reflect user behavior. The damping factor is a probability value, which is usually set to 0.85 because PageRank assumes that users have an 85% probability of following the link chain, and a 15% probability of jumping to a new page. In equation 5.1, the PageRank value of a page is the sum of the PageRank values of pages that refer to this page. Each page equally propagates its rank value to related pages.

Figure 5.2 shows an example of PageRank propagation. Page A with rank value 30 assigns a PageRank of 15 to pages B and C. Similarly, page D assigns a PageRank of 20 to B and C. However, a problem arises from the fact that the rank score of the previous page is equally distributed, without considering the meaning of the links. This feature may cause meaningless pages with many in-links to be highly ranked in the search lists.

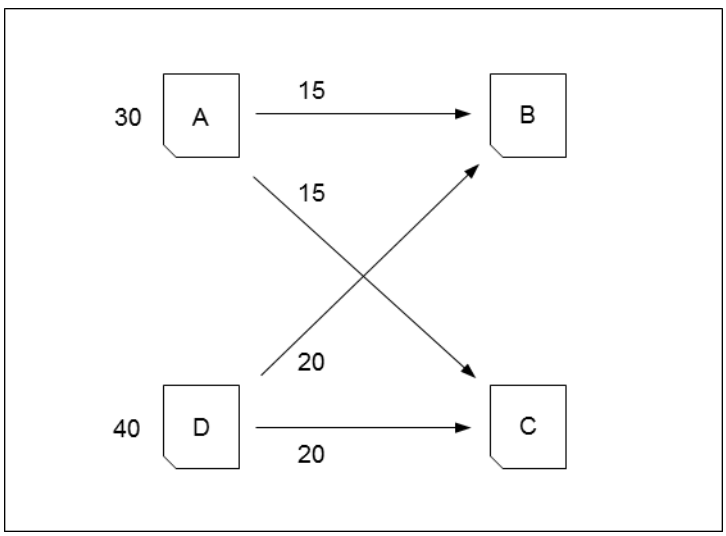



**Figure 5.2.** PageRank example.

Weighted PageRank [77] is an alternative method to avoid the uniform rank value distribution without considering the meaning of links. Weighted PageRank stratifies the distribution of the rank values based on the link weights (Figure 5.3). Equation 5.2 indicates that link weights are calculated by the proportion of the number of in-links and out-links. However, this method still considers the number

of links recursively to evaluate the weight of a link. Furthermore, because the method estimates the importance of pages by using hyper-links, it does not always guarantee that a page contains information relevant to a user's query.

$$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p \in R(v)} I_p} \quad , \quad W^{out}_{(v,u)} = \frac{O_u}{\sum_{p \in R(v)} O_p} \tag{5.2}$$

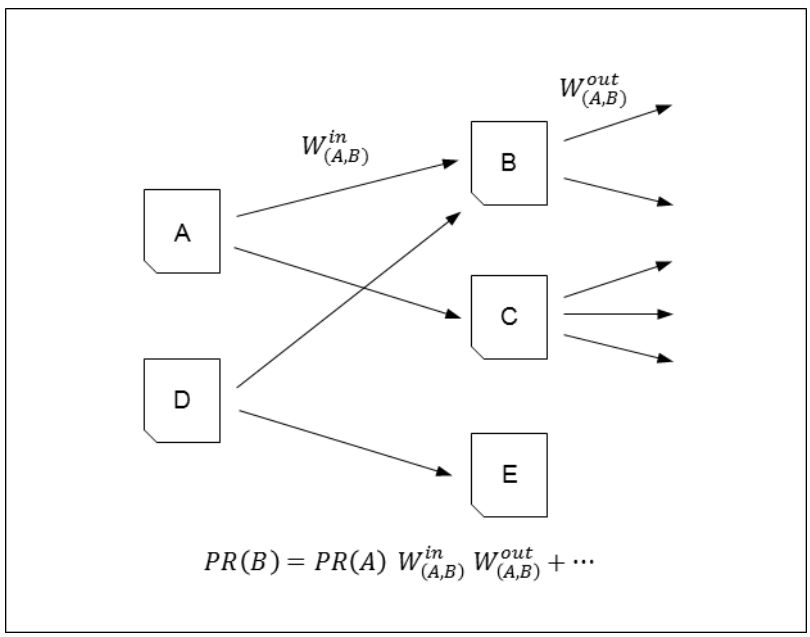



**Figure 5.3.** Weighted PageRank example.

Since Weighted PageRank, many approaches have been developed to more precisely evaluate the weight of the links. Weighted Page Content Rank [78] uses

Web content mining to improve the Weighted PageRank. Weighted Page Content Rank utilizes not only the weight of the links but also the correlation between user queries and search results. However, the computation of Weighted PageRank is still based on the number of links and does not take into account the semantic meaning of the links. Other methods, such as Topic-Sensitive PageRank [79] and personalized PageRank [80], utilize additional information. Topic-Sensitive PageRank classifies Web pages according to their topics, and computes rank values by applying a query-biased metric on the set of classified pages. Personalized PageRank is a user-biased metric that provides specific search results for each individual user. On the other hand, our primary goal is to build an integrated ranking algorithm as well as utilize semantic metadata. Thus, the scope of this thesis is set to an unbiased and explicit semantic analysis of page ranking.

## 5.3 Semantic Metadata Annotation Using RDB to RDF transformation

Semantic metadata already embedded in Web pages can be obtained by metadata parsers. If Web pages do not contain semantic metadata but contain semi-structured metadata, such as table data, extractive methods are can be used.

On the other hand, an automatic RDFa annotation system (Figure 5.4) is to generate semantic data of Web pages that do not contain any metadata. The system runs at the server side before sending Web pages to client side. In server processes,

RDB query codes are executed and query results are embedded in the Web pages. The automatic RDFa system wraps the query execution process, and generates semantic metadata of the query result based on RDB2RDF transformation.

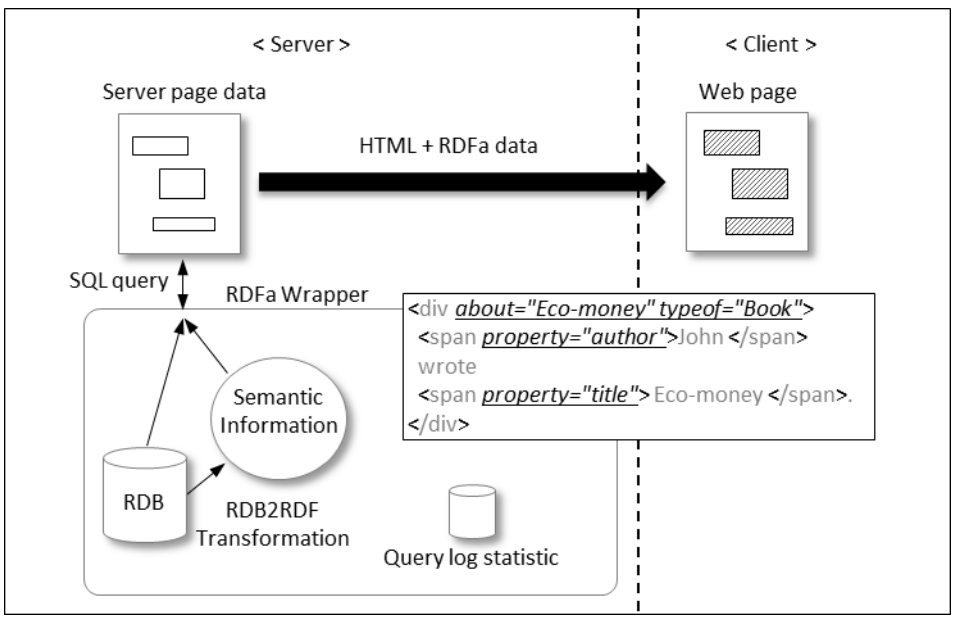



**Figure 5.4.** Automatic RDFa annotation system.

Intuitively, two query processing are performed for RDB and semantic metadata annotation: SQL query processing for RDB and SPARQL query processing for semantic data. However, the automatic annotation system provides more efficient computation method. Suppose $F$ is semantics preserving mapping function. $R$ is a relational table, where $r_i \in R$, $i$=1,2,…,$n$ ($n$ is the number of

instances in $R$). $A$ is an attribute set of $R$, where $a_j \in A$, $j$=1,2,…,$m$ ($m$ is the number of attributes in $R$). $V$ is an attribute value set of $R$, where $v_{ij} \in V$ and $r_i(a_j) = v_{ij}$. $Q$ is a SQL query for $R'(A')$ and $Q'$ is a SPARQL query identical to $Q$. For simplification of the proof, assume $R' = \{r_1, r_2\}$, $A'=\{a_1, a_2\}$, a primary key of $R$ is $a_{pk}$ that $a_{pk}=a_1$. $Q = \Pi_{a1,a2}\sigma_{a1\neq 0}(R)=\{(v_{11},v_{12}), (v_{21},v_{22})\}$. $Q'=\{t(S_{r1}, P_{a1}, O_{v11}), t(S_{r1}, P_{a2}, O_{v12}), t(S_{r2}, P_{a1}, O_{v21}), t(S_{r2}, P_{a2}, O_{v22})\}$. We can acquire $\{S_{r1}, S_{r2}\}$ using $F(R')$ in O(n), $\{P_{a1}, P_{a2}\}$ from $F(A')$ using $F(A')$ in O(n), and $\{O_{v11}, O_{v12}, O_{v21}, O_{v22}\}$ using $Q$. Therefore, SPARQL query processing can be avoided by using $Q$ and $F$ with time complexity $T(Q) + O(n)$ instead of $T(Q) + T(Q')$.

## 5.4 Information Retrieval Based on Weighted Semantic Resource Rank

In this section, a Web information retrieval method using semantic information as metadata is proposed. The proposed ranking method, called Weighted Semantic PageRank (WSPR), provides a page importance computation method based on a semantic Web data structure. WSPR directly computes the weight of the links by evaluating their meaning. WSPR is composed of four procedures (Figure 5.5). The first two procedures are related to the transformation of an existing Web structure to a semantic Web data structure. The other procedures compute rank values based on the semantic Web data structure built in the previous procedures.

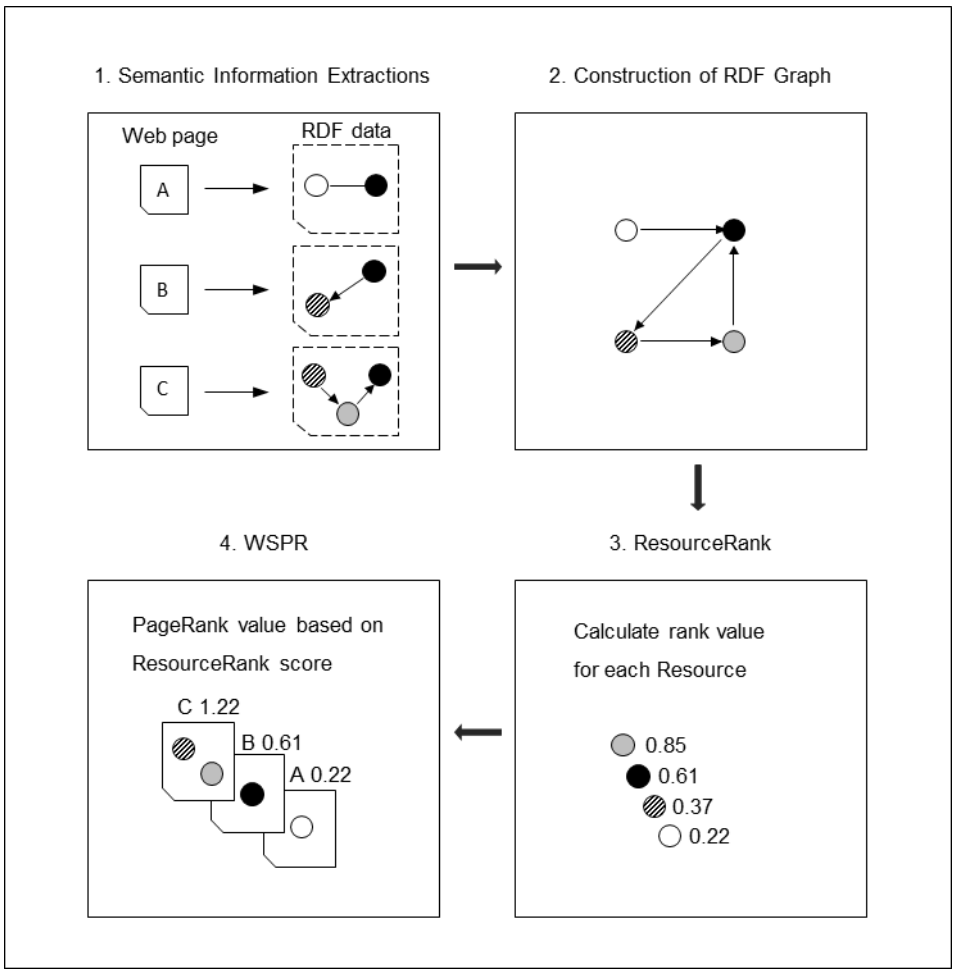



**Figure 5.5.** Overview of WSPR system.

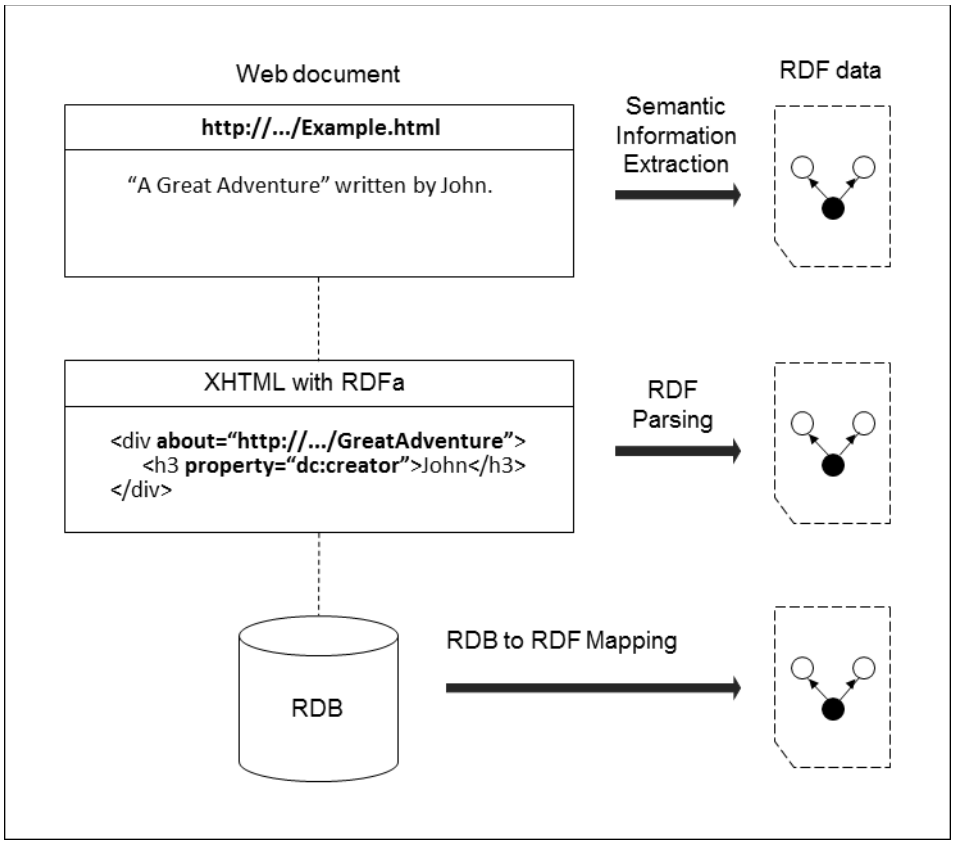



**Figure 5.6.** Semantic information extraction from existing Web data.

## 5.4.1 Semantic Information Extraction

In the first process of WSPR, the system extracts RDF metadata from the Web pages. Semantic information extraction can be processed by various methods (Figure 5.6). Manual mapping or text summarization methods generate semantic data from text data of Web documents. If Web pages are annotated by using RDFa, then the system can collect semantic metadata from Web pages. RDB to RDF transformation methods mentioned in the previous chapters can also be used to extract semantic metadata. In WSPR, collected RDF resources are used as the unit of rank values, and predicates between resources are viewed as labeled links to determine the resources' degree of importance.

## 5.4.2 Construction of an RDF Graph

In the second process, WSPR integrates the RDF dataset from the previous process into a single structure. To interconnect multiple RDF datasets, the system matches resources with a Uniform Resource Identifier. Uniform Resource Identifier (URI) is a string of characters to uniquely identify objects in the semantic Web. Figure 5.7 shows an example of RDF data integration. The system finds resources with the same URI (the black nodes in Figure 5.6 as an example), and joins the matched resources into one resource. After the procedure, all resources are connected with one another, based on the URI. The combined graph is viewed as a semantic Web data structure to be used for the evaluation of semantic resource ranks in the next process.

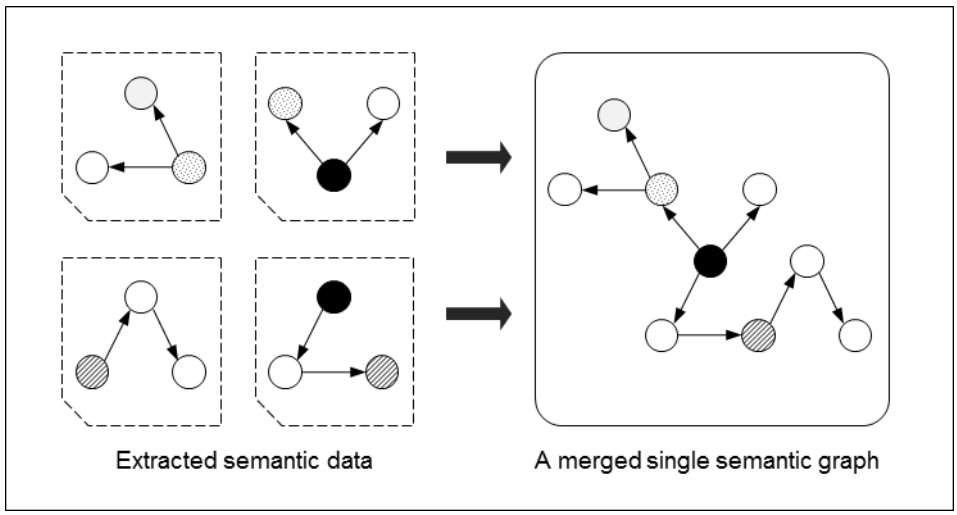



**Figure 5.7.** Merging RDF triples with resources having the same URI.

## 5.4.3 ResourceRank

In the third process, which is called ResourceRank, begins the rank evaluation of resources. ResourceRank calculates the rank values using the semantic Web data structure generated in the previous process. As the semantic Web data structure is a graph structure that edges are semantic property of RDF triples, ResourceRank evaluates the weight of the links using semantically labeled predicates among the resources. ResourceRank outputs the weights values of all semantic links. The weight values calculated by the process are able to stratify the distribution of resource rank values. Ranking resources by the weight values of links reflects the degree of semantic relationships among resources.

The weight of the links is calculated either manually or automatically [84]; this thesis is focused on generating an automatic link-weight computation metric. As of the third process, ResourceRank evaluates predicates that are links in a semantic Web data structure.

First, ResourceRank calculates the Predicate Frequency (PF) of the semantic Web data structure (Equation 5.3). The Predicate Frequency uses a function f that returns the raw frequency of a predicate. The Predicate Frequency is also normalized, by dividing by the maximum raw predicate frequency in the resource. Moreover, WSPR uses the Inverse Predicate Frequency (IPF) for balancing Predicate Frequency values (Equation 5.4)

$$PF(p,r) = \frac{f(p,r)}{\max\{f(w,r): w \in r\}} \; and \tag{5.3}$$

$$IPF(p,R) = log \frac{|R|}{|\{r \in R: p \in r\}|}, \tag{5.4}$$

where $p$ is a target predicate to compute the weight value, $r$ is a resource, and $R$ is a set of resources.

Then, the weight of the links is defined by Equation 5.5:

$$weight(r_i,p) = PF(r_i,p) \times IPF(r_i,p). \tag{5.5}$$

Finally, the ResourceRank equation takes on the form,

$$RR(r_i) = d \sum_{j \in outlink(i)} \frac{RR(r_j) \cdot weight(r_j, p)}{\sum_{j \in outlink(i)} weight(r_j, p)} + (1 - d), \qquad (5.6)$$

where $RR(r_j)$ is the ResourceRank value of resource $r_j$ that has out-link to resource $r_i$. Before summation of $RR(r_j)$ values, each $RR(r_j)$ is weighted using a value calculated by Equation 5.5. Therefore, the value of $RR(r_i)$ is a summed value by using semantically adjusted $RR(r_j)$ values.

## 5.4.4 Weighted Semantic PageRank

The final process of WSPR is the computation rank values of pages. The page rank values are calculated based on the resource rank values from the previous process. Resource rank values are returned to the pages that respectively contain the resources. It means that the resource importance is used to evaluate the importance of the pages that contain the resources. In other words, the reputation of a page is measured by the importance of the semantic resources that the page contains, rather than the number of links in the page. It improves the limitations of the link-weight evaluation methods that are based on the number of links. It also guarantees that if a page has a higher rank value, then the page contains semantically important

information. Thus, the problem that meaningless pages receive high rank values occurs lower than previous approaches.

Equation 5.7 shows the equation for the PageRank score, based on the ResourceRank scores computed in the previous procedure.

$$WSPR(p_i) = \sum_{r \in p_i} RR(r), \tag{5.7}$$

where $RR(r)$ is the ResourceRank value of resource $r$, which is contained in page $p_i$. Finally, the WSPR value of page $p_i$ is the summation of all ResourceRank values of resources in page $p_i$.

## 5.4.5 Implementation

The Hadoop framework [69] is an open source implementation of Google's MapReduce framework. Hadoop provides the Hadoop Distributed File System (HDFS) that distributes and manages large datasets over multiple servers. Hadoop MapReduce is used to perform parallel computations on Hadoop clusters. Using the MapReduce framework, researchers are able to concentrate more on implementing their algorithm and less on the parallel processing elements.

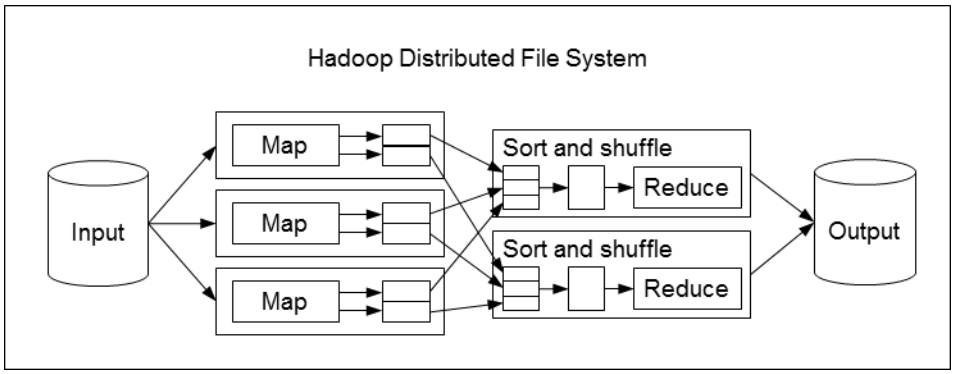



**Figure 5.8.** Overview of Hadoop MapReduce.

A MapReduce job consists of two components, a mapper and a reducer (Figure 5.8). In the map phase, input data is converted into key-value pairs. The key-value pairs are sent to the reduce phase by keys. In the reduce phase, the output dataset is generated by applying computations to the pairs received from the map phase. The MapReduce algorithm is based on the concept of map/reduce in functional programming. The method is simple and powerful, as the function runs with fault tolerant feature on parallel and distributed systems.

WSPR is implemented on Hadoop to account for the large-scale semantic metadata. The WSPR MapReduce algorithm consists of three jobs (Figure 5.9). The first job receives Web pages with semantic information as input data, and calculates the ResourceRank values of resources in the input data. After calculating the ResourceRank values, the first job outputs the result to the next job (Figure 5.10).

The second job receives the RDF resource information with ResourceRank values to compute the WSPR value. The ResourceRank scores of resources are assigned to each page where the resources were originally contained. The WSPR value of a page is calculated by summing the ResourceRank values assigned to the page (Figure 5.11).

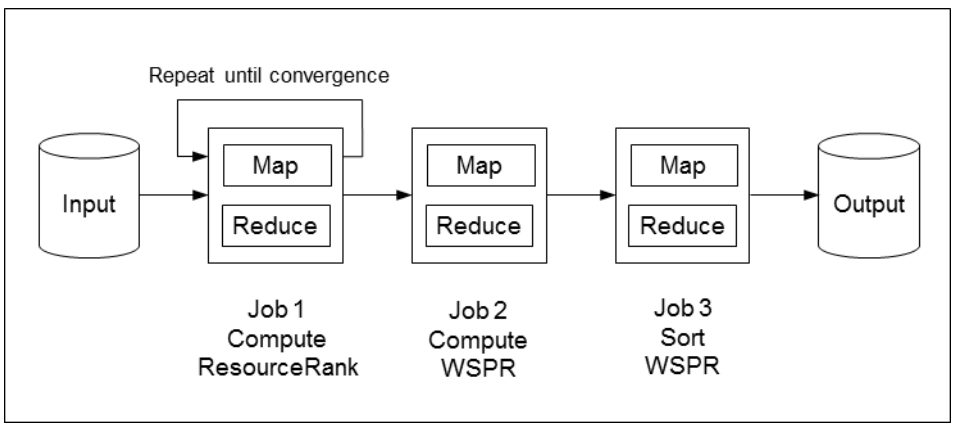



**Figure 5.9.** WSPR MapReduce job framework.

The intermediate ranking result from the second job is sent to the third job to be ordered by WSPR values. Finally, the third job outputs the page ranking result, based on the WSPR values (Figure 5.12).

```
class MAPPER
    method MAP(pageid i, page P)
        EMIT(pageid i, page P)      // Emit adjacency list
        for all pageid j ∈ P.AdjacencyList do
            r ← j.ResourceRank × j.LinkWeight
            EMIT(pageid j, r)              // Emit value for ResourceRank
        end

class REDUCER
    method REDUCE(pageid i, values [v1, v2, ...])
        R ← ∅
        sum ← 0
        for all v ∈ values [v1, v2, ...] do
            if IsResourceRankScore(v) then
                sum ← sum + v              // Sum of values for ResourceRank
            else
                R.AdjacencyList ← v    // Get adjacency list information
            end
        end
        R.ResourceRank ← sum × 0.85 + 0.15    // Compute rank
        EMIT(pageid i, page R)
```

**Figure 5.10.** MapReduce Job 1: ResourceRank.

```
class MAPPER
    method MAP(pageid i, page P)
        EMIT(pageid i, P.resourceRank)

class REDUCER
    method REDUCE(pageid i, resourceRanks [r1, r2, ...])
        R ← ∅
        sum ← 0
        for all r ∈ resourceRanks [r1, r2, ...] do
            sum ← sum + r              // ResourceRank value summation
        end
        R.PageRank ← sum
        EMIT(pageid i, page R)
```

**Figure 5.11.** MapReduce Job 2: WSPR.

```
class MAPPER
    method MAP(pageid i, page P)
        EMIT(P.PageRank, pageid i)    // Sort using Reduce function
```

**Figure 5.12.** MapReduce Job 3: Ordering pages by rank scores.

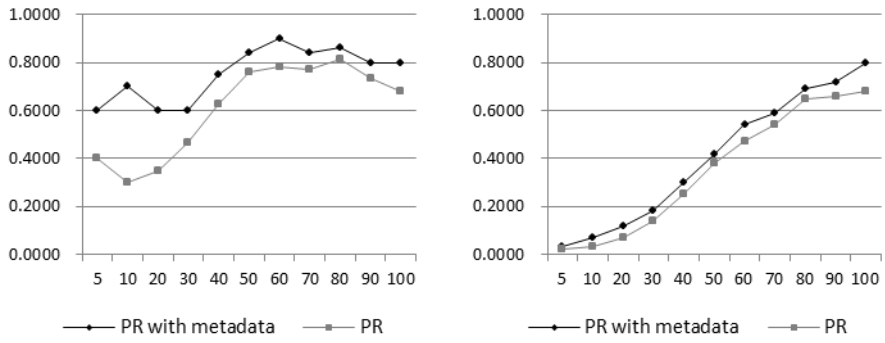
# 5.5 Evaluation

## 5.5.1 Experimental Setup

The experiments are performed on twelve nodes. One node served as the master node, while the other 11 were slave nodes. Each node had a 3.1 GHz quad-core CPU, 4GB memory, and 2TB hard disk. The operating system was 32-bit Ubuntu 12.04.2. Hadoop version 1.2.1 is used, running on Java 1.6.0. The experiments are conducted using 80,000 Wikipedia [85] Web pages with open structured schema data. As a source of Web data, the Web pages are extracted into 500,000 RDF metadata.

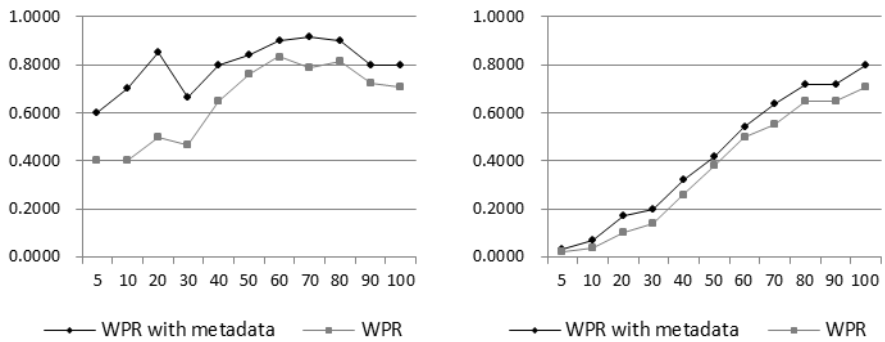
## 5.5.2 Experimental Results

In Figures 5.13 to 5.15, the results for the ranking methods on the uniform page dataset are presented. Each graph corresponds to the page ranking performance. The horizontal axis represents the size of the page set. The vertical axis in each graph is the (a) precision and (b) recall. Provided enhanced method is used to calculate ranking values based on metadata with three methods: PageRank (PR), Weighted PageRank (WPR), and Topic-Sensitive PageRank (TPR). It shows that ranking methods using metadata provides fewer false positive and false negative ranking results.



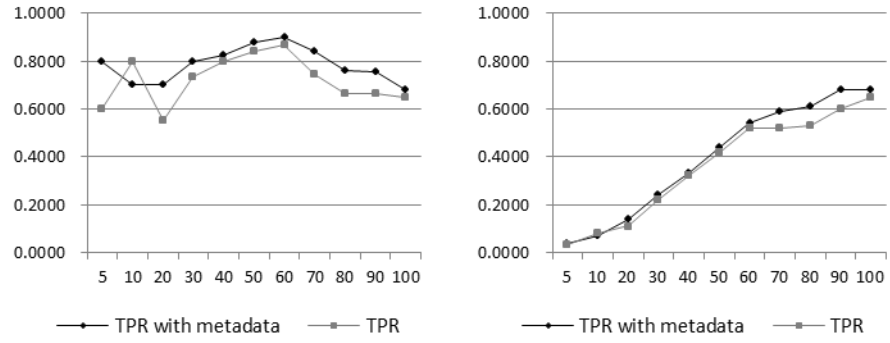

(a) Precision (b) Recall

**Figure 5.13.** Comparison of accuracy between PR and PR with metadata.




(a) Precision (b) Recall

**Figure 5.14.** Comparison of accuracy between WPR and WPR with metadata.




(a) Precision (b) Recall

**Figure 5.15.** Comparison of accuracy between TPR and TPR with metadata.

Table 5.1 shows a more detailed view of the ranking results for a query about literature. In the process of ResourceRank computation, WSPR extracted two resources, "Macmillan" and "Publishing company," from the page written about "Macmillan." The ResourceRank values for the two resources are 1.118 and 0.429, respectively. In addition, WSPR extracted one resource, "United State," with ResourceRank value 1.272 from the page written about "United States." Although the ResourceRank value of "United State" is higher than those of the other resources, the WSPR value for "United States" is lower than the other page's WSPR value (Table 5.2). It is reasonable to suppose that the page about "Macmillan," which is a publishing company, is more related to literature than the page about nations.

**Table 5.1.** ResourceRank values within pages.

| RDF Resource | ResourceRank Score |
|---|---|
| “United State” | 1.272 |
| “Macmillan” | 1.118 |
| “Publishing company” | 0.429 |

**Table 5.2.** Summary of ResourceRank used to compute WSPR.

| Page | RDF Resource (ResourceRank Score) | WSPR Score |
|---|---|---|
| Macmillan | “Publishing company” (0.429)<br>“Macmillan” (1.118) | 1.547 |
| United States | “United State” (1.272) | 1.272 |

In summary, for the problem of importance value distribution, WSPR performed better than the other methods when the pages contained semantic information. WSPR obtains semantic resources from pages in a semantic Web data structure and calculates resource rank values using the link weights among resources. Thus, WSPR is able to evaluate the rank values of Web pages based on the importance of semantic resources. The experiment showed the effectiveness of the ranking method using semantic metadata.

# Chapter 6
# Conclusions and Future Work

## 6.1 Conclusions

Issues on semantic data publication and semantic data utilization were covered in this dissertation. In particular, the concept of relational databases to RDF transformation was adapted as of improving semantic data publication, and semantic information retrieval was adapted as an advancement of semantic data utilization.

Semantics preserving RDB to RDF transformation using hierarchical direct mapping was described in Chapter 3. A metric is defined to quantify semantics preservation and is used to identify problems that occur during a mapping process. To generate sound and precise semantic data, a semantic vocabulary that contains

the hierarchical ontologies of relational data is devised. The rule sets are also defined based on the semantic vocabulary to transform relational tables and attributes. The rule sets of integrity constraints use blank nodes to group triples of relational integrity constraints. The rule based mapping system is implemented using the MapReduce framework for large-scale data. Experimental results show that the proposed method generates fewer incorrect semantic data. While preserving semantics, the method provides smaller semantic data with lesser failure rate.

Optimized Hierarchical direct mapping was described in Chapter 4. In the chapter an optimized hierarchical direct mapping and advanced semantic vocabularies were proposed. As relational attributes can be assigned by multiple integrity constraints, improved mapping rules are devised to regulate the integrity constraint output data generation. First, the rules for mapping multi-column key constraints are defined. The rules cover primary key, foreign key, unique constraint when the constraints comprise two or more attributes. Second, the rules for mapping multiple foreign keys and unique constraints are defined. Unlike multi-column primary key, two or more foreign keys or unique constraints can be contained in a single relational table, and these multiple constraints are a major factor of producing repetitive constraint output data. The improved rules guarantees that the mapping results contains only one constraint definition for one multiple key constraint. Finally, the relational meta-schema vocabulary is defined. The vocabulary initially defines relational concepts and relationships among the relational concepts. Moreover, existing mapping rules are optimized by using the

vocabulary to generate compact and intuitive semantic relational output data. The results of the evaluation show that the proposed approach generates significantly lesser number of output data while remaining semantics of original input data.

Semantic metadata based information retrieval was described in Chapter 5. A ranking method to improve the evaluation of link weights called weighted semantic pages rank (WSPR) was presented. WSPR uses semantic metadata in Web pages and adjusts the rank value propagation based on the evaluation of semantic links to obtain a more accurate ranking result. To utilize the semantic information, a hyperlink-based Web structure was transformed into a semantic Web data structure. The method effectively calculated the weight of the links to semantic resources and evaluated the importance of pages based on how many important resources they contained. Thus, in the method, pages that contained important information related to a given query could be properly ranked highly, and the probability of providing highly scored meaningless pages was lowered. The comparative evaluation of WSPR against established baseline methods clearly demonstrated the acceptable performance of the provided method.

# 6.2 Future Work

## 6.2.1 Monotone Direct Mapping

In the direct mapping, monotonicity and semantics preservation are properties that conflict with each other. If foreign key constraints are considered, a monotone direct mapping cannot be a direct mapping. As a result, existing direct mapping methods are focused on the semantics preservation than monotonicity. Nevertheless, data synchronization between original Web relational data and transformed semantic data is important for the semantic Web vision. A naïve approach to maintain monotonicity is to process RDB2RDF transformation again. However, the naïve method consumes more execution time as it processes every data including unchanged data. Thus, an investigation is required to build an effective direct mapping that can apply modified, newly created, or deleted relational data into semantic data. In that respect, it is believed that the improved direct mapping method would be advantageous to the performance of relational data to RDF transformation.

## 6.2.2 Semantic Data Utilization Using Topic Model

The estimation of the relevance between users query and topics of Web documents is crucial to retrieve accurate query results. The proposed method explained in Chapter 5 improves the limitation of existing ranking method, which

cannot evaluate the importance of links by its meaning as hyper-links do not have explicit referencing comments. However, the proposed method only considers evaluation of semantic data contained in Web documents. To calculate a more relative to users query, utilization of other meta-information would be needed. First, analysing user profiles or Web exploration history can be better filtering information to provide user-specific retrieval results. However, the method contains additional challenging issues about personal privacy. Second, learning topics of Web documents can also improve the performance of information retrieval methods. As Web documents usually do not explicitly provide its topics, various topic models are used to extract topics of Web documents. In view of information retrieval, topic models are useful tools to find semantic topics of Web pages. Therefore, future directions to explore include using suitable methods to evaluate semantic topic information based on topic models.

# Appendix A
# Proofs

## A.1 Proof of Lemma 1

**Lemma 1:** Suppose $R$ is a relational table set, $A$ is an attribute set, $K$ is an integrity constraint set, $I$ is a relational instance set, $X$ is a set where $X \subset (R \cup A \cup K \cup I)$, F is a direct mapping function. Then, we can retrieve every $y \in F(X)$ by inference from owl:Class.

**Proof:** If $x \in R$, then $x$ can be transformed directly into owl:Class. If $x \in A$, then $x$ can be transformed into either owl:ObjectProperty or owl:DatatypeProperty, which are types of rdfs:Class. If $x \in A \cup K$, then $x$ can be transformed into owl:FunctionalProperty or owl:InverseFunctionalProperty, which are types of rdfs:Class. If $x \in K$, then $x$ can be transformed into owl:onProperty, owl:minCardinality, owl:maxCardinality, or owl:cardinality, all of which are types of rdf:Property, as well as the type of rdf:Property is rdfs:Class. As rdfs:Class is a subclass of owl:Class, semantic resources used in transformation are directly or indirectly assigned to owl:Class type. Therefore, transformed semantic data F($x$) can be retrieved by inference using owl:Class.

# A.2 Proof of Lemma 2

**Lemma 2:** For any $X$ in relational data, if $x \in X$ references another $y \in X$, then $x$ can be transformed into a semantic resource, which has a type of owl:ObjectProperty.

**Proof:** Let *a* is an attribute and *b* is a relational table. If *a* is a foreign key of table *x* referencing a primary key attribute in table *y*, then *a* can be transformed into owl:ObjectProperty with domain *x* and range *y*. If *b* is a binary relation over table *t* and *u*, then *b* can be transformed into owl:ObjectProperty with domain *t* and range *u*. Even if bigger relationship exists, the relation can be transformed using owl:ObjectProperty. Therefore, owl:ObjectProperty can describe any referencing relationship among relational data sources.

## A.3 Proof of Lemma 3

**Lemma 3:** Suppose *G* is an RDF graph, $G_1$ and $G_2$ are components of *G*, there is no edge between $G_1$ and $G_2$, $G_1$ is rooted at $x \in R$, $G_2$ is rooted at $y \in R$, where *R* is a set of semantic resources, then,

(1) If *x* and *y* have the same URI, then *x* is identical with *y*. Thus, $G_1$ and $G_2$ can be merged into one graph.

(2) If $x$ and $y$ have different URIs, $x$ has a property $p_1$, and $y$ has a property $p_2$ that has the same URI as $p_1$, then $G_1$ and $G_2$ cannot be merged into one graph, and $p_1$ can be distinguished from $p_2$ using $x$ and $y$.

**Proof:** First, we prove (1). Assume that $G_1$ and $G_2$ cannot be merged into one graph, $G_1$ has only one triple t($x$, $p_1$, $o_1$), $G_2$ has only one triple t($y$, $p_2$, $o_2$) that is identical to t($x$, $p_2$, $o_2$), and q($s$, $p$, $o$) is a query function to find triples. If q($x$, *?p*, *?o*) is the input of the query function, then the result is { t($x$, $p_1$, $o_1$), ($x$, $p_2$, $o_2$) }. This contradicts our assumption that $G_1$ and $G_2$ cannot be merged into one graph. Therefore, if $x$ and $y$ have the same URI, then $G_1$ and $G_2$ can be merged into one graph. Second, we prove (2). Assume that $G_1$ and $G_2$ can be merged into one graph, $p$ is a property that has the same URI as $p_1$ and $p_2$; $p = p_1 = p_2$, $G_1$ has only one triple t($x$, $p$, $o_1$), $G_2$ has only one triple t($y$, $p$, $o_2$), and q($s$, $p$, $o$) is a query function to find triples. If q($y$, $p$, $o_2$) is the input of the query function, then there is no matching result. If q($x$, $p$, $o_1$) is the input for the query function, then the result is also empty. This contradicts our assumption that $G_1$ and $G_2$ can be merged into one graph. Therefore, if $x$ and $y$ have different URI, then $G_1$ and $G_2$ cannot be merged into one graph.

# A.4 Proof of Lemma 4

**Lemma 4:** Consider X is a set of relational data, F is an RDB to RDF mapping function, and G is an RDF to RDB inverse-mapping function of F. If mapping rules are defined based on Lemma 1, 2, and 3, then,

**(1) Soundness:** the mapping rules are sound that the rules generate only semantics in RDB data ($X \supseteq G(F(X))$).

**(2) Completeness:** the mapping rules are complete that the rules generate all semantics in RDB data ($X \subseteq G(F(X))$).

**Proof 4.1:** First, we prove the completeness of the mapping rules by induction. Suppose S is an RDB schema set, S' is semantic graph data that represents every schema in S, F is a mapping function and G is an inverse-function of F, and X is input data, where $X \subset S$. (1) Base: Assume X = { }, then F(X) = { } and G(F(X)) = { }, thus $X \subseteq G(F(X))$. (2) Inductive hypothesis: Given a set $X = \{x_1, x_{2,\ldots,}x_k\}$ and |X| = k. Assume F(X) is a mapping function for all $X \subset S$, which transforms X to semantic data that $F(X) \subset S'$. (3) Inductive step: given a set of RDB data $\{x_1, x_{2,..} x_{k+1}\}$ and |X| = k+1. Consider the set X' without x, where x is nay element of X, and |X'| = k. We have X' and x that is excluded from X, then we can apply the inductive hypothesis to X', mapping that they all be transformed into semantic data $F(X') \subset Y$ and $G(F(X')) \subset S$. We can also apply the inductive

hypothesis to x, which is identical to X when k is 1. Therefore, by the mapping rules, $X \subseteq G(F(X))$.

**Proof 4.2:** Second, we prove the soundness of the mapping rules. Assume S is an RDB schema set, S' is semantic graph data that represents every schema in S, F is a mapping function that generates a directed graph with a root node containing a unique URI value, and G is an inverse-mapping function of F. If input data X is a disjoint set, $|X| = k$, $X = \{x_1, x_2, \ldots, x_k\}$, then $F(X) = y_k$ is a graph data, F(X) is a disjoint set by lemma 3.

# A.5 Proof of Theorem 1

**Theorem 1:** The provided rules are sound and complete for the semantics preserving RDB to RDF transformation.

**Proof:** From Lemma 4.1, it follows that RDB to RDF mapping rules defined by Lemma 1, 2, and 3 generate only semantics in RDB data. From Lemma 4.2, it follows that RDB to RDF mapping rules defined by Lemma 1, 2, and 3 generate all semantics in RDB data. For this point, by theorem 4, the provided rules are sound and complete.

# Appendix B
# Semi-automatic Semantic Data Publication

A semi-automatic RDFa annotation system (Figure A.1) is to generate semantic data of Web pages that do not contain any metadata. The system receives Web pages as input and matches the pages to semantic resources of linked open data. The system also provides a method to manually annotate semantic metadata. Finally, the system generates semantic data-annotated Web pages as output.

The semi-automatic RDFa annotation system includes the following three components:

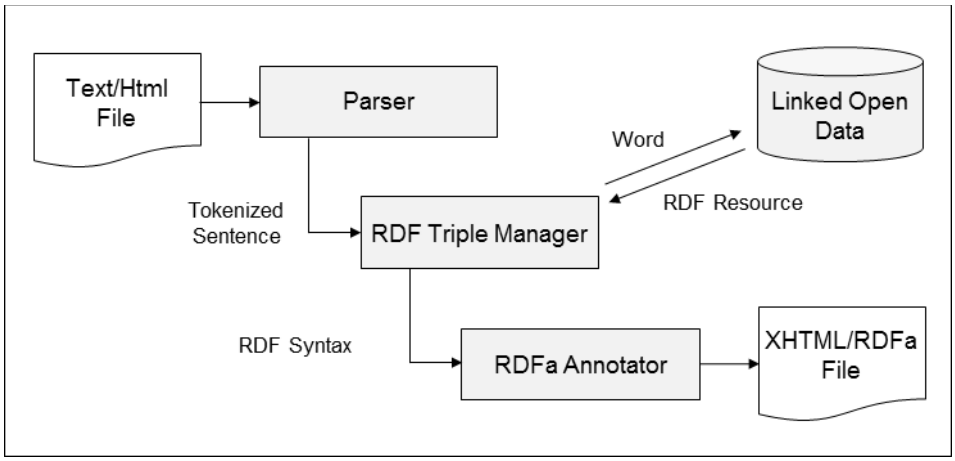



**Figure A.1.** Semi-automatic RDFa annotation system.

Input Box: The system reads the input data, parses the data sentence by sentence, and tokenizes each sentence into words. Then, the parsed data is moved to an RDF triple management procedure.

RDF Triple Manager: The system finds candidate resources that match the words from Linked Open Data, such as DBPedia [87]. The user can then choose an appropriate resource from among the candidates. The user places words in the input box as the subject of a triple. In addition, the system provides a set of predicates from RDF vocabularies for interoperability.

Result Code View: The system generates RDFa meta tags based on the information provided by the previous step. Finally, the resource code view shows a Web document with RDFa annotations.

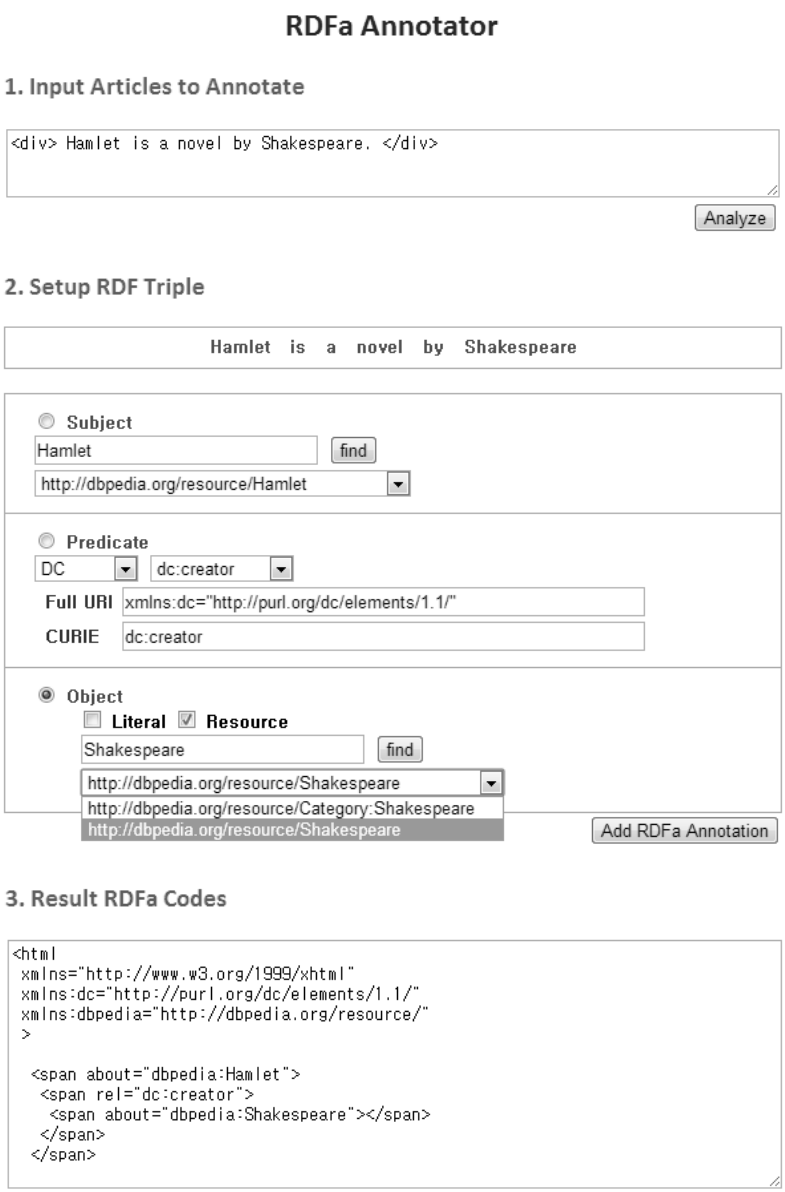



**Figure A.2.** User interface of the provided annotation system.

# Appendix C

# Specifications

## C.1 Hierarchical Semantic Vocabulary

| | |
|---|---|
| Classes | Relation |
| Properties | Attribute \| NonFKeyAttr \| FKeyAttr \| BinaryRelation \| IdentifyingRelationship \| NonIdentifyingRelationship |

```
@prefix : <http://idb.snu.ac.kr/rdb2rdf/hsv/>.
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<!-- HSV 1 -->
:Relation  rdf:type   owl:Class .

<!-- HSV 2 -->
:Attributerdf:type   rdf:Property .
:Attributerdf:type   owl:FunctionalProperty .
:Attributerdfs:domain          :Relation .

:NonFKeyAttr        rdf:type   owl:DataTypeProperty .
:NonFKeyAttr        rdfs:subClassOf      :Attribute .

:FKeyAttr rdf:type   owl:ObjectProperty .
:FKeyAttr rdf:subClassOf       :Attribute .
:FKeyAttr rdfs:range           :Relation .

<!-- HSV 3 -->
:BinaryRelation      rdf:type   owl:ObjectProperty .
:BinaryRelation      rdfs:domain          :Relation .
:BinaryRelation      rdfs:range           :Relation .

<!-- HSV 4 -->
:IdentifyingRelationship       rdf:type   owl:ObjectProperty .
```

```
:IdentifyingRelationship        rdfs:domain          :Relation .
:IdentifyingRelationship        rdfs:range           :Relation .


<!-- HSV 5 -->
:NonIdentifyingRelationship  rdf:type  owl:ObjectProperty .
:NonIdentifyingRelationship  rdfs:domain          :Relation .
:NonIdentifyingRelationship  rdfs:range           :Relation .
```

# C.2 Relational Meta-Schema Vocabulary

| | |
|---|---|
| Classes | Relation \| ConstrainedRelation \| NotNullConstrainedRelation \| UniqueConstrainedRelation \| PKConstrainedRelation \| FKConstrainedRelation \| DefaultConstrainedRelation \| CheckConstrainedRelation |
| Properties | Attribute \| NonFKeyAttr \| BinaryRelation \| IdentifyingRelationship \| NonIdentifyingRelationship \| NotNullConstrainedAttribute \| UniqueConstrainedAttribute \| UniqueConstrainedAttribute \| PKConstrainedAttribute \|<br><br>DefaultConstrainedAttribute \| CheckConstrainedAttribute \| defaultValue \| checkCondition \| hasConstraint \| composedOf |

```
@prefix : <http://idb.snu.ac.kr/rdb2rdf/rmsv/>.
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<!-- RMSV 1 -->
:Relation  rdf:type   owl:Class .
:ConstrainedRelation            rdf:type   :Relation .

:NotNullConstrainedRelation rdf:type   :ConstrainedRelation .
:UniqueConstrainedRelation  rdf:type   :ConstrainedRelation .
:PKConstrainedRelation      rdf:type   :ConstrainedRelation .
:FKConstrainedRelation      rdf:type   :ConstrainedRelation .
:DefaultConstrainedRelation rdf:type   :ConstrainedRelation .
:CheckConstrainedRelation   rdf:type   :ConstrainedRelation .

<!-- RMSV 2 -->
:Attributerdf:type   rdf:Property .
:Attributerdf:type   owl:FunctionalProperty .
:Attributerdfs:domain          :Relation .

:NonFKeyAttr        rdf:type   owl:DataTypeProperty .
:NonFKeyAttr        rdfs:subClassOf      :Attribute .

:ConstrainedAttribute          rdf:type   :Attribute .
```

```
<!-- RMSV 3 -->
:BinaryRelation      rdf:type   owl:ObjectProperty .
:BinaryRelation      rdfs:domain          :Relation .
:BinaryRelation      rdfs:range           :Relation .

<!-- RMSV 4 -->
:IdentifyingRelationship       rdf:type   owl:ObjectProperty .
:IdentifyingRelationship       rdfs:domain          :Relation .
:IdentifyingRelationship       rdfs:range           :Relation .

<!-- RMSV 5 -->
:NonIdentifyingRelationship    rdf:type   owl:ObjectProperty .
:NonIdentifyingRelationship    rdfs:domain          :Relation .
:NonIdentifyingRelationship    rdfs:range           :Relation .


<!-- RMSV 6: Constrained Relation -->
:NotNullConstrainedRelation rdfs:subClassOf     :_bNotNullConstrainedRelation .
:_bNotNullConstrainedRelation          rdf:type   owl:Restriction .
:_bNotNullConstrainedRelation          owl:onProperty     :NotNullConstrainedAttribute .
:_bNotNullConstrainedRelation          owl:Cardinality    1 .


:UniqueConstrainedRelation  rdfs:subClassOf     :_bUniqueConstrainedRelation .
:_bUniqueConstrainedRelation           rdf:type   owl:Restriction .
:_bUniqueConstrainedRelation           owl:onProperty     :UniqueConstrainedAttribute .
:_bUniqueConstrainedRelation           owl:maxCardinality 1 .
```

```
:PKConstrainedRelation          rdfs:subClassOf      :_bPKConstrainedRelation .
:_bPKConstrainedRelation        rdf:type   owl:Restriction .
:_bPKConstrainedRelation        owl:onProperty       :PKConstrainedAttribute .
:_bPKConstrainedRelation        owl:Cardinality      1 .

:FKConstrainedRelation          rdfs:subClassOf      :_bFKConstrainedRelation .
:_bFKConstrainedRelation        rdf:type   owl:Restriction .
:_bFKConstrainedRelation        owl:onProperty       :FKConstrainedAttribute .
:_bFKConstrainedRelation        owl:minCardinality 1 .

:DefaultConstrainedRelation rdfs:subClassOf      :_bDefaultConstrainedRelation .
:_bDefaultConstrainedRelation           rdf:type   owl:Restriction .
:_bDefaultConstrainedRelation           owl:onProperty       :DefaultConstrainedAttribute .
:_bDefaultConstrainedRelation           owl:maxCardinality 1 .

:DefaultConstrainedRelation rdfs:subClassOf      :_bCheckConstrainedRelation .
:_bCheckConstrainedRelation rdf:type   owl:Restriction .
:_bCheckConstrainedRelation owl:onProperty       :CheckConstrainedAttribute .
:_bCheckConstrainedRelation owl:maxCardinality 1 .



<!-- RMSV 7: Constrained Attribute -->
:NotNullConstrainedAttribute rdf:type   :ConstrainedAttribute
:UniqueConstrainedAttribute rdf:type   :ConstrainedAttribute
:UniqueConstrainedAttribute rdf:type   owl:InverseFunctionalProperty
:PKConstrainedAttribute         rdf:type   :ConstrainedAttribute
:PKConstrainedAttribute         rdf:type   owl:InverseFunctionalProperty
```

```
:DefaultConstrainedAttribute rdf:type   :ConstrainedAttribute
:defaultValue        rdf:type   owl:DataTypeProperty
:defaultValue        rdfs:domain        :DefaultConstrainedAttribute
:CheckConstrainedAttribute   rdf:type   :ConstrainedAttribute
:checkCondition      rdf:type   owl:DataTypeProperty
:checkCondition      rdfs:domain        CheckConstrainedAttribute
:checkCondition      rdfs:range         xsd:string



<!-- RMSV 8 Constraint Assertion -->
:hasConstraint       rdfs:subPropertyOf rdf:type
:composedOf          rdf:type   owl:ObjectProperty
:composedOf          rdfs:domain        :ConstrainedRelation
:composedOf          rdfs:range         :ConstrainedAttribute
```

# Bibliography

[1] RDF Working Group. (2004). RDF 1.1 Primer [Online]. Available: https://www.w3.org/TR/rdf11-primer/ [Accessed: October 2016].

[2] RDF Working Group. (2004). RDF Schema 1.1 [Online]. Available: https://www.w3.org/TR/rdf-schema/ [Accessed: October 2016].

[3] OWL Working Group. (2004). OWL Web ontology language [Online]. Available: https://www.w3.org/TR/owl-ref/ [Accessed: October 2016].

[4] He B., Patel M., Zhang Z., and Chang K. C. C., "Accessing the deep web," Communications of the ACM, vol. 50, no. 5, pp. 94-101, 2007.

[5] De Laborda C. P. and Conrad S., "Database to Semantic Web mapping using RDF query languages," In: International conference on conceptual modeling. Springer Berlin Heidelberg, pp. 241-254, 2006.

[6] Spanos D. E., Stavrou P., and Mitrou N., " Bringing relational databases into the semantic web: A survey," Semantic Web, vol 3, no. 2, pp. 169-209, 2012.

**Bibliography**

[7] T. Berners-Lee. (1998). Relational Databases on the Semantic Web [Online]. Available: http://www.w3.org/DesignIssues/RDB-RDF.html [Accessed: October 2016].

[8] Arenas M., Bertails A., Prud'hommeaux E., and Sequeda J. (2012). A Direct Mapping of Relational Data to RDF [Online]. Available: http://www.w3.org/TR/rdb-direct-mapping/ [Accessed: October 2016].

[9] Sequeda, J. F., Arenas, M., and Miranker, D. P., "On directly mapping relational databases to RDF and OWL," In Proceedings of the 21st international conference on World Wide Web. ACM, pp. 649-658, 2012.

[10] Sequeda J. F., Arenas M., and Miranker D. P., "A completely automatic direct mapping of relational databases to RDF and OWL," In: Proceedings of posters and demos at the 10th international semantic web conference (ISWC), 2011.

[11] Kobayashi M. and Takeda K., "Information Retrieval on the Web," ACM Computing Surverys, vol. 32, no. 2, pp. 144-173, 2000.

[12] Gudivada V. N., Raghavan V. V., Grosky W. I., and Kasanagottu R., "Information Retrieval on the World Wide Web," IEEE Internet Computing, vol. 1, no. 5, pp. 58-68, 1997.

[13] Singhal A., "Modern Information Retrieval: A Brief Overview," Bull. IEEE Comput. Soc. Tech. Comm. Data Eng., vol. 24, no. 4, pp. 35-43, 2001.

[14] Brin S. and Page L., "Reprint of: The Anatomy of a Large-Scale Hypertextual Web Search Engine," Comput. Networks, vol. 56, no. 18, pp. 3825-3833, 2012.

[15] Kleinberg J. M., "Authoritative Sources in a Hyperlinked Environment," J. ACM, vol. 46, no. 5, pp. 604-632, 1999.

[16] W3C. Semantic Web. [Online]. Available: https://www.w3.org/standards/semanticweb/ [Accessed: October 2016].

[17] Carol J. J. and Klyne, G., "Resource Description Framework: Concepts and Abstract Syntax," W3C Recommendation, 2004.

[18] W3C. (2001). URIs, URLs, and URNs: Clarifications and Recommendations 1.0 [Online] Available: https://www.w3.org/TR/uri-clarification/ [Accessed: October 2016].

[19] Gutierrez C., Hurtado C., and Mendelzon A. O., "Foundations of semantic web databases," In Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM., pp. 95-106, 2004.

[20] Gutierrez C., Hurtado C. A., and Vaisman A., "Introducing time into RDF," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 2, pp. 207-218, 2007.

[21] W3C. (2004). RDF 1.1 XML Syntax [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/ [Accessed: October 2016].

[22] W3C. (2008). Notation3 (N3): A readable RDF syntax [Online]. Available: https://www.w3.org/TeamSubmission/n3/ [Accessed: October 2016].

[23] W3C. (2014). RDF 1.1 Turtle [Online]. Available: https://www.w3.org/TR/turtle/ [Accessed: October 2016].

[24] W3C. (2008). SPARQL Query Language for RDF [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/ [Accessed: October 2016].

[25] W3C. (2012). R2RML: RDB to RDF Mapping Language [Online]. Available: https://www.w3.org/TR/r2rml/ [Accessed: October 2016].

[26] W3C Working Group. (2014) RDFa Core 1.1 - Third Edition. [Online]. Available: http://www.w3.org/TR/rdfa-syntax/ [Accessed: October 2016].

[27] Khare R., "Microformats: The Next (Small) Thing on the Semantic Web?," IEEE Internet Comput., vol. 10, no. 1, pp. 68-75, 2006.

[28] W3C Working Group. (2011). HTML Microdata [Online]. Available: http://www.w3.org/TR/2011/WD-microdata-20110405/. [Accessed: October 2016].

[29] Steiner T. and Hausenblas M., "How Google is Using Linked Data Today and Vision For Tomorrow," Linked Data Futur. Internet Futur. Internet Assem. (FIA 2010), Ghent, Belgium, 2010.

[30] Facebook. (2014). The Open Graph Protocol [Online]. Available: http://ogp.me. [Accessed: October 2016].

[31] Corlosquet S., Cyganiak R., Polleres A., and Decker S., "RDFa in Drupal: Bringing Cheese to the Web of Data," Proc. 5th Work. Scripting Dev. Semant. Web ESWC, 2009.

[32] De Virgilio R., Frasincar F., Hop W., and Lachner S., "A Reverse Engineering Approach for Automatic Annotation of Web Pages," Multimed. Tools Appl., vol. 64, no. 1, pp. 119-140, 2013.

[33] Duma M., "RDFa Editor for Ontological Annotation," Proc. Student Res. Work. Assoc. with RANLP, pp. 54-59, 2011.

[34] Samwald M., Lim E., Masiar P., Marenco L., Chen H., Morse T., Mutalik P., Shepherd G., Miller P., and Cheung K. H., "Entrez Neuron RDFa: A Pragmatic Semantic Web Application for Data Integration in Neuroscience Research," Proc. Int. Conf. Eur. Fed. Med. Informatics, pp. 317-321, 2009.

[35] Khalili A., Auer S., and Hladky D., "The RDFa Content Editor - From WYSIWYG to WYSIWYM," Proc. IEEE Signat. Conf. Comput. Software, Appl. COMPSAC, pp. 531-540, 2012.

[36] Tramp S., Heino N., Auer S., and Frischmuth P., "RDFauthor: Employing RDFa for Collaborative Knowledge Engineering," Proc. Cimiano, P., Pinto, H.S. EKAW 2010. LNCS, vol. 6317, pp. 90-104, 2010.

[37] W3C. (2009). OWL 2 Web Ontology Language Document Overview [Online]. Available: https://www.w3.org/TR/2009/WD-owl2-overview-20090327/ [Accessed: October 2016].

[38] Lassila O. and Swick R. R. Resource description framework (RDF) model and syntax specification. 1999.

[39] Dau F. RDF as graph-based, diagrammatic logic. In: international symposium on methodologies for intelligent systems, pp. 332-337, 2006.

[40] Hayes J. (2004). A Graph Model for RDF [Online]. Available: http://users.dcc.uchile.cl/~cgutierr/papers/rdfgraphmodel.pdf [Accessed: October 2016].

[41] Berners-Lee T. and Connoly D. (2004). Delta: an ontology for the distribution of differences between RDF graphs [Online]. Available: http://www.w3.org/DesignIssues/Diff [Accessed: October 2016].

[42] Sahoo S. S., Halb W., Hellmann S., et al., "A survey of current approaches for mapping of relational databases to RDF," W3C RDB2RDF Incubator Group Report, 2009.

[43] Michel F., Montagnat J., Faron-Zucker C., "A survey of RDB to RDF translation approaches and tools," Research report, I3S 2014.

[44] Byrne K., "Having triplets-holding cultural data as RDF," In: Proceedings of the ECDL 2008 workshop on information access to cultural heritage, 2008.

[45] Green J., Dolbear C., Hart G., et al., "Creating a semantic integration system using spatial data," In: Proceedings of the 2007 international conference on posters and demonstrations, vol. 401, pp. 70-71, 2008.

[46] RDB2RDF Working Group. (2012). R2RML: RDB to RDF mapping language [Online]. Available: https://www.w3.org/TR/r2rml/ [Accessed: October 2016].

[47] Hert M., Reif G., and Gall H. C., "A comparison of RDB-to-RDF mapping languages," In: Proceedings of the 7th international conference on semantic systems. ACM, pp. 25-32, 2011.

[48] Bizer C., and Seaborne A., "D2RQ-treating non-RDF databases as virtual RDF graphs," In: Proceedings of the 3rd international semantic web conference (ISWC), 2004.

[49] Bizer C., Cyganiak R., Garbers J., Maresch O., and Becker C. (2009). The D2RQ Platform v0.7 - Treating Non-RDF Relational Databases as Virtual RDF Graphs - User Manual and Language Specification [Online]. Available: http://wifo5-03.informatik.uni-mannheim.de/bizer/d2rq/spec/20090810/ [Accessed: October 2016].

[50] Erling O. and Mikhailov I., "RDF support in the Virtuoso DBMS," In: Networked Knowledge-Networked Media. Springer Berlin Heidelberg, pp. 7-24, 2009.

[51] OpenLink Software. (2007). Mapping Relational Data to RDF with Virtuoso [Online]. Available: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/ VOSSQLRDF [Accessed: October 2016].

[52] Sequeda J. F., Depena R., and Miranker D. P., "Ultrawrap: using sql views for RDB2RDF," In: Proceedings of the international conference on semantic systems, 2009.

[53] Sequeda J. F. and Miranker D. P., "Ultrawrap: SPARQL execution on relational data," Web semantics: science, services and agents on the World Wide Web 22, pp. 19-39, 2013.

[54] RDB2RDF Working Group. (2012). A Direct Mapping of Relational Data to RDF [Online]. Available: https://www.w3.org/TR/rdb-direct-mapping/ [Accessed: October 2016].

[55] Strandhaug M., "An R2RML Mapping Management API in Java: Making an API Independent of its Dependencies," 2014.

[56] Hert M., Reif G., and Gall H. C., "A comparison of RDB-to-RDF mapping languages," In Proceedings of the 7th International Conference on Semantic Systems ACM, pp. 25-32, 2011.

[57] Hogan A., Arenas M., Mallea A., and Polleres A., "Everything you always wanted to know about blank nodes," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 27, pp. 42-69, 2014.

[58] Mallea A., Arenas M., Hogan A., and Polleres A., "On blank nodes," In: International semantic Web conference (ISWC). Springer Berlin Heidelberg, pp. 421-437, 2011.

[59] Tirmizi S. H., Sequeda J., and Miranker D., "Translating sql applications to the semantic web," In: International conference on database and expert systems applications. Springer Berlin Heidelberg, pp. 450-464, 2008.

[60] Du H., Wery L., "Micro: A normalization tool for relational database engineers," Journal of Network and Computer Applications, vol. 22, no. 4, pp. 215-232, 1999.

[61] Wang S., Shen J., Hong T., "Mining fuzzy functional dependencies from quantitative data," In: IEEE International Conference on Systems, Man and Cybernetics, 2000.

[62] Li M., Du X. Y., and Wang S., "Learning ontology from relational database," In: 2005 International Conference on Machine Learning and Cybernetics. IEEE, vol. 6, pp. 3410-3415, 2005.

[63] Shen G., Huang Z., Zhu X., and Zhao X., "Research on the Rules of Mapping from Relational Model to OWL," In: Proceedings of the Workshop on OWL: Experiences and Directions (OWLED), 2006.

[64] Astrova I., Korda N., and Kalja A., "Rule-based transformation of SQL relational databases to OWL ontologies," In: Proceedings of the 2nd International Conference on Metadata & Semantics Research, 2007.

[65] Cullot N., Ghawi R., and Yetongnon K., "DB2OWL: a tool for automatic database-to-ontology mapping," In: Italian symposium on advanced database systems (SEBD), pp. 491-494, 2007.

[66] Cerbah F., "Learning highly structured semantic repositories from relational databases," In: European semantic web conference (ESWC) Springer Berlin Heidelberg, pp. 777-781, 2008.

[67] Lim K. B., Jun H. G., and Kim H. J., "Semantics preserving MapReduce process for RDB to RDF transformation," International journal of metadata, semantics and ontologies, vol. 10, no. 4, pp. 229-239, 2015.

[68] Sequeda J. F., Tirmizi S. H., Corcho O., and Miranker D. P., "Survey of directly mapping SQL databases to the Semantic Web," The knowledge engineering review, vol. 26, no. 4, pp. 445-486, 2011.

[69] Apache Software Foundation. Apache Hadoop [Online]. Available: http://hadoop.apache.org [Accessed: October 2016].

[70] Dean J. and Ghemawat S., "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.

[71] Ensembl Compara [Online]. Available: www.ensembl.org/info/docs/api/compara [Accessed: October 2016].

[72] Ensembl [Online]. Available: www.ensembl.org [Accessed: October 2016].

[73] phpMyAdmin [Online]. Available: https://www.phpmyadmin.net [Accessed: October 2016].

[74] MusicBrainz [Online]. Available: https://musicbrainz.org [Accessed: October 2016].

[75] DBT2 [Online]. Available: http://osdldbt.sourceforge.net [Accessed: October 2016].

[76] Stein K. and Hess C., "Information Retrieval in Trust-Enhanced Document Networks," Semant. Web Min., pp. 65-81, 2006.

[77] Xing W. and Ghorbani A., "Weighted PageRank Algorithm," Proc. Second Annu. Conf. Commun. Networks Serv. Res. CNSR 2004, pp. 305-314, 2004.

[78] Sharma P., Tyagi D., and Bhadana P., "Weighted Page Content Rank for Ordering Web Search Result," Int. J. Eng. Sci. Technol., vol. 2, no. 12, pp. 7301-7310, 2010.

[79] Haveliwala T. H., "Topic-Sensitive PageRank," Proc. 11th Int. Conf. World Wide Web, pp. 517-526, 2002.

[80] Jeh G. and Widom J., "Scaling Personalized Web Search," Proc. twelfth Int. Conf. World Wide Web WWW, pp. 271-279, 2003.

[81] RDF Working Group. (2004). Resource Description Framework [Online]. Available: http://www.w3.org/RDF/. [Accessed: October 2016].

[82] Bayardo R. J., Ma Y., and Srikant R., "Scaling up all Pairs Similarity Search," Proc. 16th Int. Conf. World Wide Web, pp. 131-140, 2007.

[83] Salton G. and Buckley C., "Term-weighting Approaches in Automatic Text Retrieval," Inf. Process. Manag., vol. 24, no. 5, pp. 513-523, 1988.

[84] Toupikov N., Umbrich J., Delbru R., Hausenblas M., and Tummarello G., "DING! Dataset RankING Using Formal Descriptions," Proc. WWW 2009 Work. Linked Data Web (LDOW 2009), Madrid, Spain, 2009.

[85] Wikipedia. [Online]. Available: http://en.wikipedia.org/. [Accessed: October 2016].

[86] Jarvelin K. and Kekalainen J., "Cumulated Gain-based Evaluation of IR Techniques," ACM Trans. Inf. Syst., vol. 20, no. 4, pp. 422-446, 2002.

[87] Bizer C., Lehmann J., Kobilarov G., Auer S., Becker C., Cyganiak R., and Hellmann S., "DBpedia - A Crystallization Point for the Web of Data," J. Web Semant. Sci. Serv. Agents World Wide Web, vol. 7, no. 3, pp. 154-165, 2009.

[88] W3C. (2009). RDFS 3.0 [Online]. Available: http://www.cs.vu.nl/~guus/public/rdfs30-v01.html/ [Accessed: December 2016].

[89] Allemang D. and James H., "Semantic web for the working ontologist: effective modeling in RDFS and OWL," Elsevier, 2011.

[90] Buccella A., Penabad M. R., Rodriguez F. J., Farina A., and Cechich A., "From relational databases to OWL ontologies," In Proceedings of the 6th National Russian Research Conference, 2004.

# 초 록

다이렉트 매핑은 W3C에 의해 권고된 대표적인 RDB2RDF 변환 방법이다. 다이렉트 매핑은 자동 변환 규칙을 사용하여 RDB 데이터를 RDF 데이터로 변환하는 방법으로, 정보 손실 없는 변환을 뜻하는 '의미 보존'을 매우 중요한 속성으로 다루고 있다. 그러나 현재까지의 방법은 특정 상황에서 의미 보존을 달성하지 못하는 문제를 보이고 있다. 따라서 본 논문은 계층적 방법을 제안하여 다이렉트 매핑의 의미 보존 속성을 이룬다. 계층적 다이렉트 매핑 규칙은 의미 데이터 변환의 특징에 대한 보조정리에 기반하여 정의되었으며, 보다 명료하고 적합한 의미 데이터의 생산을 위해 계층적 의미 어휘 집합을 사용해 RDB2RDF 변환을 수행한다. 두 번째로, RDB2RDF 변환 시 원본 데이터에 기술된 제약조건들이 반복적으로 생성되는 현상을 줄인 최적화된 계층적 다이렉트 매핑 방법을 제안한다. 특히 한 관계형 테이블 안에 다중 컬럼 키 제약 조건이 선언 되어 있거나 다수의 외래키 혹은 고유키가 선언된 경우 동일한 제약 조건에 대한 의미 정보가 반복되어 생성되는 문제를 해결하기 위해, 제약 조건과 다중 컬럼 선언 부분을 분리하여 처리하는 방법으로 문제를 해결한다. 더 나아가 관계형 데이터의 개념 및 속성을 기술한 어휘 집합을 활용해 보다 경량화되고 직관적으로 이해 가능한 의미정보를 생성하는 변환 방법을 제안한다. 마지막으로 RDB2RDF 데이터 변환으로 생성된 의미 정보에 기반한 랭킹 방법을 제안한다. 웹 데이터에 대한 의미 정보를 추출하고, 생성된 RDF 그래프의 의미 기반 링크로 가중치를 계산하여 웹 문서들의 중요도를

결정한다. 본 논문에서는 실험 결과를 통해 제안한 방법이 중복 생성을 줄이고 정보 손실 없이 의미 정보를 생성하며 보다 효과적으로 의미 정보 검색을 수행함을 보인다.

**주요어:** 시맨틱웹, 데이터베이스, RDB2RDF, RDF, RDFS, OWL, 의미정보검색

**학 번:** 2011-30253